Improving the quality of software translation

Timothy O. Hassall

Computing with Management (Industry)

Session 2006/2007

The candidate confirms that the work submitted is their own and the appropriate credit has been given where reference has been made to the work of others.

I understand that failure to attribute material which is obtained from another source may be considered as plagiarism.

(Signature of student)

Summary

This report looks to examine in detail the process of translating software into different foreign languages, and attempts to improve on it. It focused in particular on the process used by Xerox GKLS.

The various software translation/localisation tools available on the market were examined and literature on the subject of translation was read to understand the various quality issues that affected the industry. Field research was carried out to get the views of the translators and others working in localisation. From this it was decided that an application to visualise software in the TRADOS translation environment, as well as improving upon TRADOS' comment system, were appropriate means on improving the process.

The implementation of the project involved a great deal of interaction with TRADOS' own software development kit, regular expressions, the Component Object Model (COM) and XML.

Evaluation of the finished product included usability aspects and an attempt to measure improvements in translation quality, decided upon based on research of machine translation evaluation techniques.

Acknowledgements

I would like to thank all those people who have been involved in the project from Xerox GKLS for their input and time in their already busy schedules.

Thank you to my supervisor Dr. Clive Souter for his invaluable input and support throughout the project.

Thank you to my assessor, Eric Atwell for his feedback on my mid project report and progress meeting.

I would also like to thank my family who have supported me greatly with this project and far beyond it.

Table of contents

1. Introduction 1
  1.1 Aim 1
  1.2 Objectives 1
  1.3 Minimum Requirements 1
  1.4 Relevance to degree 1
2 Background Research 2
  2.1 Localisation 2
    2.1.1 Translation Memory 5
    2.1.2 Trados 6
    2.1.3 Quality Issues in software localisation 8
    2.1.4 Alternative Software Localisation tools 9
  2.2 Machine Translation and evaluating translation quality 10
  2.3 Usability 11
3 Field Research 13
4. Preparation of Solution 15
  4.1 Choice of solution 15
  4.2 Methodology 15
  4.3 Requirements Gathering 16
  4.4 Project Plan 17
  4.5 Selection of Development Language 18
5. Design & Implementation 19
  5.1 Affected file types 19
    5.1.1 .resx files 19
    5.1.2 .TTX files 19
    5.1.3 TTX comment file 20
  5.2 Using the TRADOS SDK 21
    5.2.1 Implementing TRADOS Plug ins – COM 22
  5.3 Preparation tool for .resx files 23
  5.4 Visualisation of .resx TTX files 25
    5.4.1 Class design 25
    5.4.2 Regular expressions 25
    5.4.3 Approach to reading .resx.TTX files 26
    5.4.4 Optimising the TTX reading code 27
    5.4.5 Approach to measuring bounding boxes 29
    5.4.6 Hotkey checking 30
    5.4.7 The main visualisation program 30
  5.5 Extending TRADOS Comments functionality 31
    5.5.1 Generating a comment report 31
    5.5.2 Displaying comment information to translators on future projects 32
  5.6 Documentation 33
  5.7 Testing 34
    5.7.1 .resx Visualisation 34
    5.7.2 Results 35
    5.7.3 Comment Functionality 35
6. Evaluation 37
  6.1 Evaluation against requirements 37
  6.2 Evaluation of Project Management 37
  6.3 End User Evaluation 38
    6.3.1 Planning 38
    6.3.2 Results 42
  6.4 Plan for future evaluation 43
  6.5 Conclusions 43
7. Future Developments 44
    7.1 Extending the visualization plug-in 44
    7.2 Extending the comments system 44

8. Bibliography 46
Appendix A – Personal Reflections 48
Appendix B – Software Localisation Tools 49
Appendix C – Field Research Notes 52
Appendix D – Requirements Specification provided to Xerox prior to meeting 58
Appendix E – Documentation 61
Appendix G – End User Evaluation 76

1. Introduction

Xerox GKLS Language Services (LS) is a Localisation (see section 2.1) services provider with an annual turnover of $20 million. LS localises software and documentation for both internal and external clients which include major automotive, telecommunications and IT corporations. This project examines the process used to translate software and looks to improve it.

1.1 Aim

The aim of this project was to improve the quality of natural language translation of software in all foreign languages.

1.2 Objectives

The objective of this project was to develop tools within the TRADOS translation environment that would benefit the work of translators when translating software.

1.3 Minimum Requirements

1) A solution that would allow ..resx windows resource files to be viewed and translated, as if they were built software, into foreign languages in the Trados translation environment.

1.4 Relevance to degree

This project involved the understanding of software engineering concepts, gained from the SE15, 20 and 24 modules, as well as considerations of usability (GI11) in looking at existing systems and developing new ones.

2 Background Research

At Xerox GKLS there is a need to provide good quality products to ensure customer satisfaction. To ensure the quality of software translation two rounds of translation take place. The second round of translation is carried out by an experienced translator, who quality-checks the first cycle with the benefit of the re-compiled/built software. This stage can be costly, since experienced translators are expensive and engineers have to build the software and support the validation in the event that the translator encounters problems with the software. The costs of validation will vary dependent on total word count and the quality of translator used at the translation stage, but it often accounts for as much as 40% of total project cost.

2.1 Localisation

Localisation (or Localization) is defined by the Localisation Industry Standards Association (LISA) as follows:

“Localisation involves taking a product and making it linguistically and culturally appropriate to the target locale (country/region and language) where it will be used and sold.” [1]

The major process involved in localisation is the translation of the product, but there are a number of other processes involved:

- Project Management
- Engineering of software
- Desk Top Publishing (DTP) of documentation
- Functionality testing of localised software or web applications

At Xerox GKLS, for which this project took place, the usual process for localising software is as follows:

1. Software files received, files requiring translation identified and separated by Software Localisation Engineer. Software files can be in a number of different formats, such as resource files like ..resx or .rc, or actual source code files eg. .java. A section of an English ..resx file is shown in figure 2.1 and the software it is associated with in figure 2.2:

```
<data name="checkBox1.Size" type="System.Drawing.Size,
System.Drawing">
    <value>163, 18</value>
  </data>
  <data name="checkBox1.TabIndex" type="System.Int32, mscorlib">
    <value>10</value>
  </data>
  <data name="checkBox1.Text" xml:space="preserve">
    <value>Merge into existing memory?</value>
  </data>
  <data name="&gt;&gt;checkBox1.Name" xml:space="preserve">
    <value>checkBox1</value>
  </data>
  <data name="&gt;&gt;checkBox1.Type" xml:space="preserve">
    <value>System.Windows.Forms.CheckBox, System.Windows.Forms,
Version=2.0.0.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089</value>
  </data>
```

Figure 2.1 – Source English .resx file

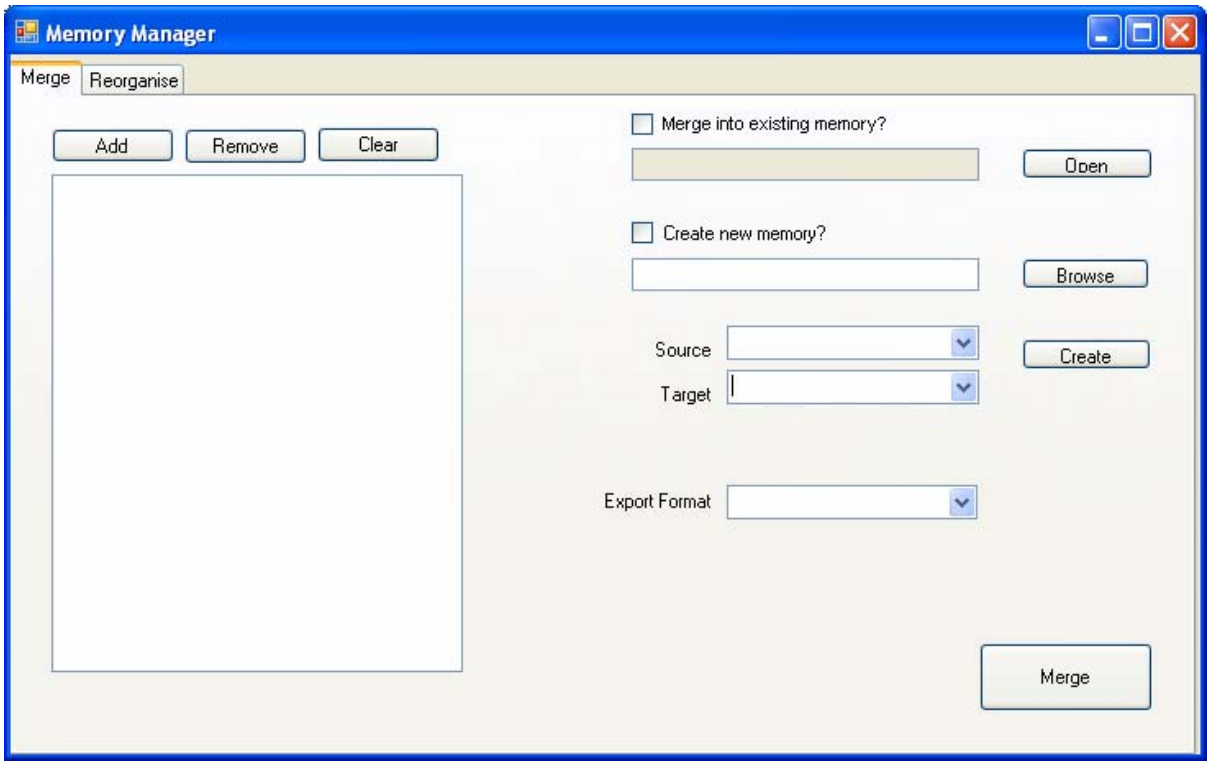


Figure 2.2 – Source English software associated with .resx file from figure 2.1

2. Files prepared for translation by TRADOS Specialist, converted into TRADOS' intermediary bilingual format (.TTX) and pre translated against translation memory (TM), if any exists (see sections 2.1.1 and 2.1.2)

3. Translators are commissioned by a project coordinator (if freelance translators are used), and TRADOS TTX files and TM are sent to them for translation. (Figure 2.6 shows the files in the Translation environment TagEditor (section 2.1.2)).
4. Translators return files, and these files are “Cleaned up” (converted to their original file format, but now with translated target strings).
5. Software localisation engineer builds the software using the newly translated files.
6. Experienced translators are employed to “validate” translation, using the built, translated software as reference.
7. Validated Trados files are “cleaned up”.
8. Final localised software is built.
9. If requested, the software is tested, and then returned to the customer. Figures 2.3 and 2.4 show translated .resx sample and software.

```
<data name="checkBox1.Size" type="System.Drawing.Size,
System.Drawing">
    <value>163, 18</value>
  </data>
  <data name="checkBox1.TabIndex" type="System.Int32, mscorlib">
    <value>10</value>
  </data>
  <data name="checkBox1.Text" xml:space="preserve">
    <value>Verleiben in Existierenerinnerung ein?</value>
  </data>
  <data name="&gt;&gt;checkBox1.Name" xml:space="preserve">
    <value>checkBox1</value>
  </data>
  <data name="&gt;&gt;checkBox1.Type" xml:space="preserve">
    <value>System.Windows.Forms.CheckBox, System.Windows.Forms,
Version=2.0.0.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089</value>
  </data>
```

Figure 2.3 –Translated .resx File

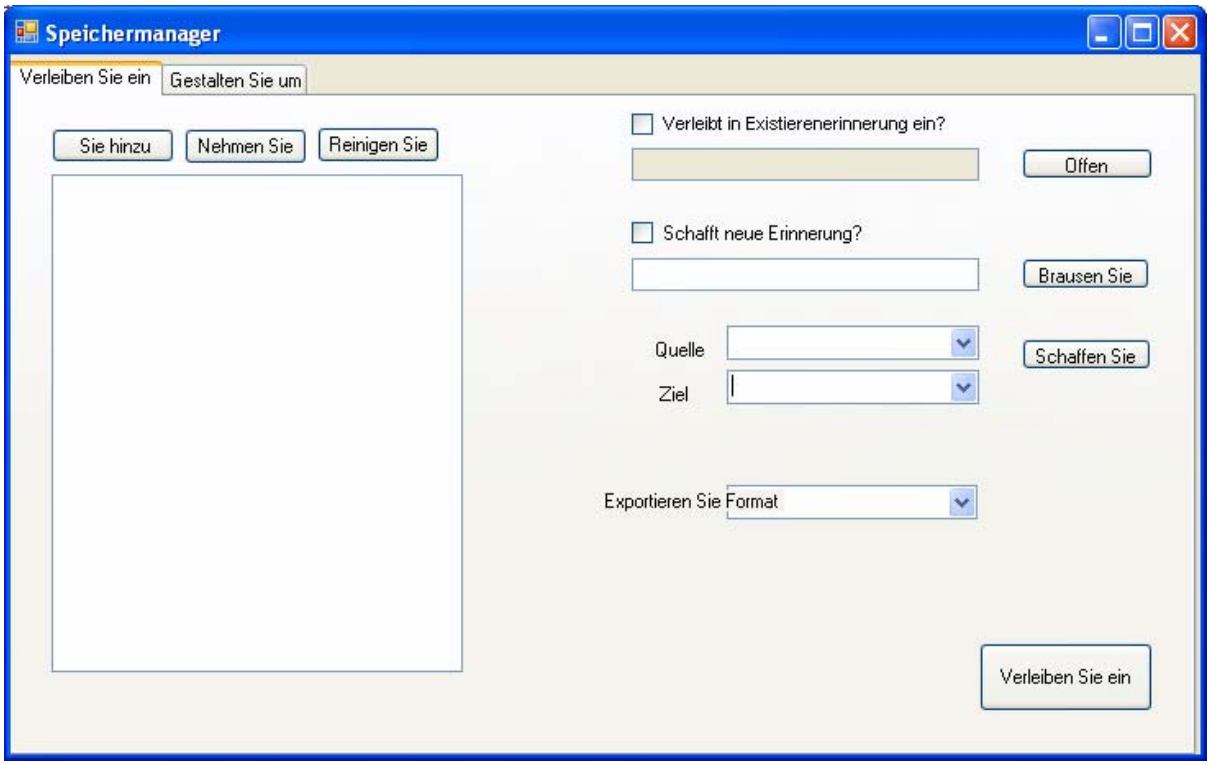


Figure 2.4 – Translated software associated with .resx file from figure 2.3

2.1.1 Translation Memory

Translation Memory is described as an aligned parallel corpus [2] – source and target segments of text, generally up to the length of a sentence.

"A translator can consult a database of previous translations, usually on a sentence-by-sentence basis, looking for anything similar enough to the current sentence to be translated, and can then use the retrieved example as a model." [3]

The translation memory is often used to 'Pre-translate' files. "Pre-translation refers to the process of comparing a complete source text to a Translation Memory (TM) database and automatically inserting the translations of all exact matches found in the database. The result is a hybrid text containing pre-translated and untranslated segments." [3]

In Trados' TM system, the user is given an indication of how close a translation in memory is to what is currently being translated. The translation will be marked with a percentage match. For example if the source text in the new file is "The printer engine is on", and there is a translation in memory for "the printer engine is off", a fuzzy match will be displayed, with a percentage match of around 85%.

TM's have become increasingly important in the industry as a mechanism for reducing the overall project costs for customers. If a piece of software has been previously localised and has only experienced a small update of perhaps 100 words out of 5000, then the customer will not be happy to pay for the whole piece of software to be translated again.

2.1.2 Trados

Trados [4] is a suite of Computer Aided Translation tools, currently on version 7.5 (also known as TRADOS 2006). The key tools are Translators Workbench, a translation memory interface, and TagEditor, an editing environment. As figure 2.6 shows, TagEditor displays the bilingual TTX format created by TRADOS.

Workbench (figure 2.5) allows the translator to access a particular translation memory, searching its contents, or being given options for a translation if used in combination with TagEditor (if there are a number of translations with potential relevance). Within workbench there are also functions to analyse the source text to gain a word count. This will measure the total word count, and show the relevance of the TM's contents to the source text - How many 100% matches there are etc..

Also part of TRADOS' suite of tools is terminology management environment called MultiTerm, and tools for pre processing Adobe FrameMaker documents. A particularly important tool is Xtranslate. This takes an existing TTX file from an earlier project, matches it against a new file for an update of that project, inserting segments where the source text and surrounding segments are identical. This ensures that the context remains unchanged. At Xerox , these segments are not checked by translators and therefore the customer is not charged.

TRADOS TagEditor has some features to speed up translation, an example of which is "Open Next No 100%". This looks for the next segment which doesn't have a 100% match from the TM associated with it, and opens it. This is especially useful in long files where much of the content has been previously translated. It also has a comment system in place, where the translator can make notes on their translations. These comments are only accessible from within the TTX file.

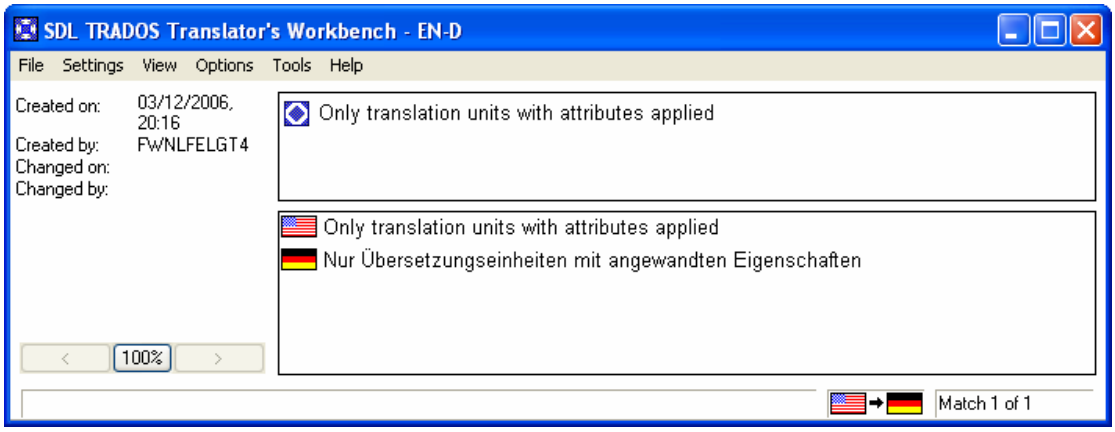

SDL TRADOS Translator's Workbench - EN-D
File Settings View Options Tools Help
Created on: 03/12/2006, 20:16
Created by: FWNLFELGT4
Changed on:
Changed by:
Only translation units with attributes applied
Only translation units with attributes applied
Nur Übersetzungseinheiten mit angewandten Eigenschaften
100%
Match 1 of 1

Figure 2.5 : Translators workbench

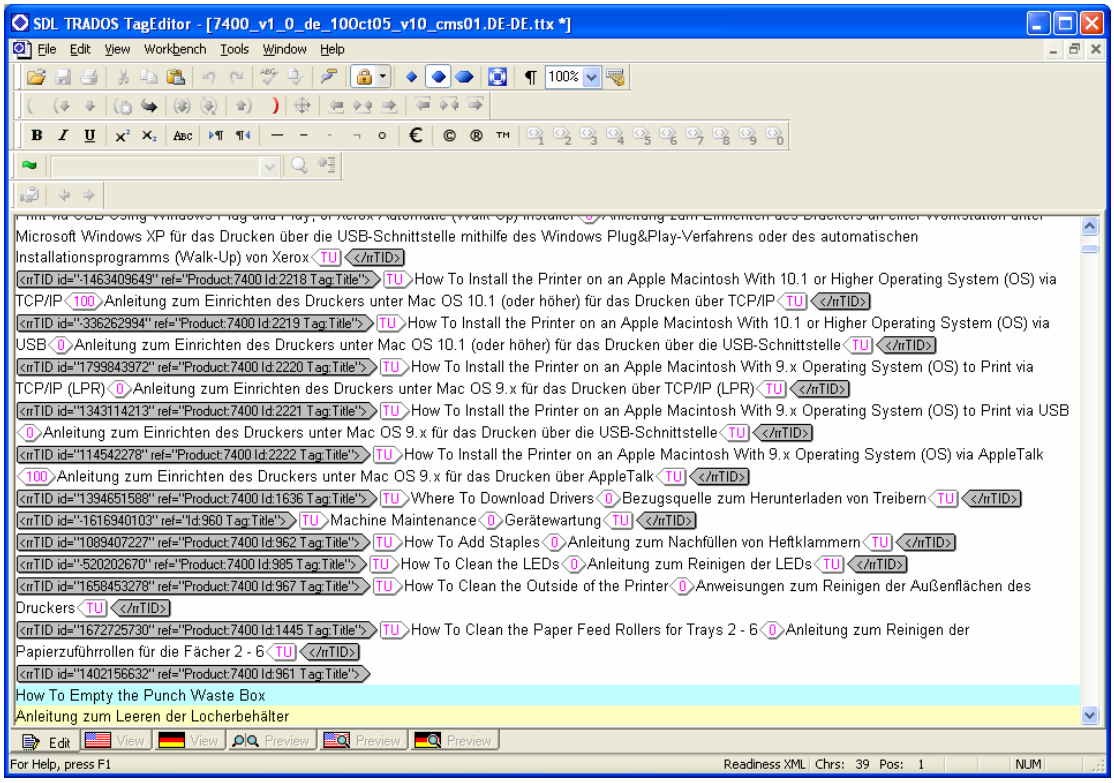

SDL TRADOS TagEditor - [7400_v1_0_de_10Oct05_v10_cms01.DE-DE.ttx *]
File Edit View Workbench Tools Window Help
Microsoft Windows XP für das Drucken über die USB-Schnittstelle mithilfe des Windows Plug&Play-Verfahrens oder des automatischen Installationsprogramms (Walk-Up) von Xerox
How To Install the Printer on an Apple Macintosh With 10.1 or Higher Operating System (OS) via TCP/IP
Anleitung zum Einrichten des Druckers unter Mac OS 10.1 (oder höher) für das Drucken über TCP/IP
How To Install the Printer on an Apple Macintosh With 10.1 or Higher Operating System (OS) via USB
Anleitung zum Einrichten des Druckers unter Mac OS 10.1 (oder höher) für das Drucken über die USB-Schnittstelle
How To Install the Printer on an Apple Macintosh With 9.x Operating System (OS) to Print via TCP/IP (LPR)
Anleitung zum Einrichten des Druckers unter Mac OS 9.x für das Drucken über TCP/IP (LPR)
How To Install the Printer on an Apple Macintosh With 9.x Operating System (OS) to Print via USB
Anleitung zum Einrichten des Druckers unter Mac OS 9.x für das Drucken über die USB-Schnittstelle
How To Install the Printer on an Apple Macintosh With 9.x Operating System (OS) via AppleTalk
Anleitung zum Einrichten des Druckers unter Mac OS 9.x für das Drucken über AppleTalk
Where To Download Drivers
Bezugsquelle zum Herunterladen von Treibern
Machine Maintenance
Gerätewartung
How To Add Staples
Anleitung zum Nachfüllen von Heftklammern
How To Clean the LEDs
Anleitung zum Reinigen der LEDs
How To Clean the Outside of the Printer
Anweisungen zum Reinigen der Außenflächen des Druckers
How To Clean the Paper Feed Rollers for Trays 2 - 6
Anleitung zum Reinigen der Papierzuführrollen für die Fächer 2 - 6
How To Empty the Punch Waste Box
Anleitung zum Leeren der Locherbehälter
Edit View View Preview Preview Preview
For Help, press F1
Readiness XML Chrs: 39 Pos: 1
NUM

Figure 2.6 : TRADOS TagEditor

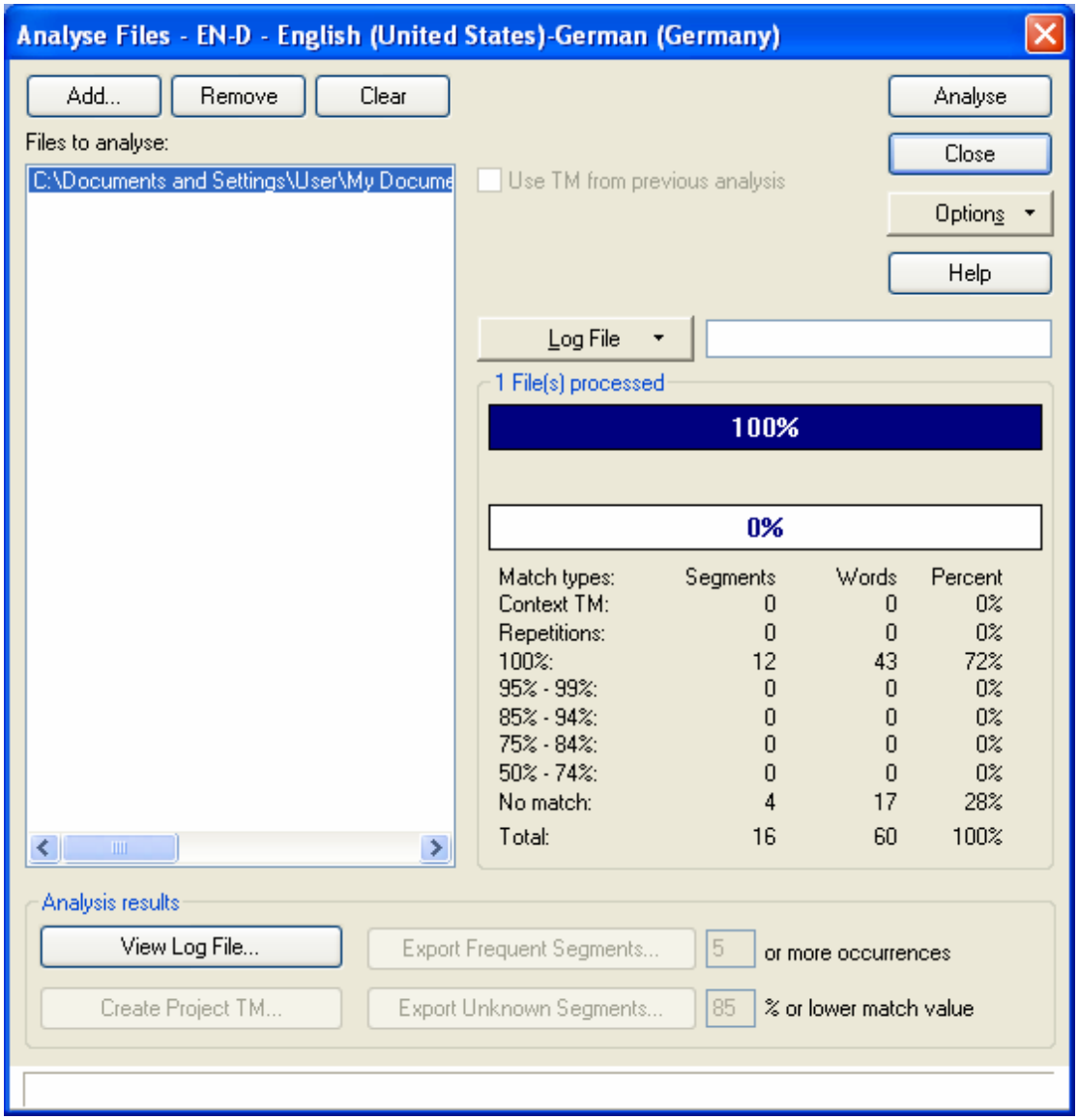


Figure 2.7 : Analyse files in workbench

2.1.3 Quality Issues in software localisation

- Bounding boxes – when software is designed, developers with internationalization in mind allow space for text expansion since foreign languages tend to be longer than English. Not taking this into account would increase the likelihood of text truncation. Even with this provision, there is still a possibility that translators will exceed the bounding box boundaries.
- Hotkeys are shortcut keys used in software. These are often selected because the key letter is part of the word of the function eg. Pressing X to use the “Exit” function. It is likely that menu option names will change during any translation process, thus hotkey assignment must be checked to ensure duplications have not been made.

- Clarity – Within TRADOS and many other translation tools, the translator will not be presented with a visual representation of the software they are required to translate. They will see a view similar to that shown in Figure 2. “Deconstructing the context….represents one of the greatest challenges for translators working today” [5]. The translator must understand the “’Topography’ of the software (from where does the specific content emanate, how are the various application data sources related to one another…” [5]. Software files often contain variables eg. %d representing a decimal integer, so if the rest of the segment to be translated does not clarify what the variable will be, the likelihood is that the translator will have to seek assistance from a terminologist or localisation engineer.
- Subject Matter – A translator with a greater understanding of the product they are translating will produce higher quality work.

2.1.4 Alternative Software Localisation tools

There are a number of tools on the market designed specifically for dealing with the localisation of software. The most prominent of these are Alchemy CATALYST [6] and Pass Engineering’s PASSOLO [7]. Both of these tools allow the user to view the software screens as if it were built and add their translations to it, in addition to handling bounding boxes, hotkeys etc.

Catalyst can handle a number of different file types, but PASSOLO is limited to .dll files. PASSOLO at one point was used on some Xerox projects, but only by engineering staff as part of a time consuming workaround – importing translations from TRADOS via a word table.

Catalyst also has some flaws. It has no means of preventing bounding box manipulation by translators. It also lacks many of the features contained in TagEditor to speed-up translation and it also doesn’t have the sophisticated TM options that TRADOS possesses.

Very few translators own such tools, as after spending £500 or more on TRADOS which can be used for documentation and web-based projects, spending more (Catalyst Translator edition costs over £300) on a tool only to be used occasionally is unattractive. The Localisation Service Provider largely dictates the choice of tool and they tend to favour the all-encompassing solutions, like TRADOS, even with their shortcomings when translating software, due to its superior TM technology and coverage of file formats. Purchasing applications like Alchemy CATALYST is also expensive for Localisation Service Providers. Each Professional License costs £5000.

The Windows Resource Localisation Editor [8] allows the user to perform many of the functions of User Interface design that are available in Visual Studio, but only requires the availability of the ..resx file. The user can change the properties of UI components as well as their text. This software appears to be more appropriate for use after the .resx file has been translated, and would more commonly be used by localisation engineers before and after translation. Prior to translation, if allowed by the customer they would make changes to bounding-boxes or UI component locations if they believe they will be particularly problematic eg. a button's bounding-box being large enough for the source English text, but too small for many other languages.

This software has no support for translation memory – any translations from the past would be lost, or at least more difficult to obtain. Also, the editor allows the translator to edit properties that they normally would not be allowed to. It may be appropriate for very small translation jobs where the translator would benefit from the visual context, and it offers the advantage of being free (it is available as part of the downloadable .NET framework), but on the whole is more useful as an engineering tool. Screenshots of all these tools can be found in Appendix B.

2.2 Machine Translation and evaluating translation quality

There has been a great deal of research into different means of evaluating machine translation, and some of these could perhaps be applied to evaluating the improvements in quality of translation of natural language in this project. The types of Machine Translation (MT) evaluation are as follows [9]:

- Feasibility – evaluation of the potential of a new MT approach
- Requirements Elicitation – Building prototypes to determine specific functions for possible implementations as part of an MT system
- Internal/Progress Evaluation – Regular evaluations of MT components prior to system release
- Diagnostic evaluation – Evaluation of functionality characteristics of prototype by researchers/developers
- Declarative evaluation – Evaluators judge MT output quality using selected metrics.
- Usability evaluation –Evaluators representative of end-users test how easy the application is to use.
- Operational evaluations – Managers calculate the purchase and running costs of an MT system and compare these with its benefits
- Comparison evaluation – Declarative, Usability and Operational evaluations combined to compare systems.

Although a usability evaluation will be performed, our main concern in this area of research will be the declarative and operational evaluation. This evaluation aims to measure the quality of translation and its financial benefits.

The most relevant of the human evaluation types in this context would be “evaluation by post editing effort”. Since there are already two stages of translation in the current process, and in that although the initial translation is by a human, less time spent on a validation would imply that quality at the first translation stage had improved.

There are also a number of automated MT evaluation approaches, which have been developed in an attempt to deal with the large amount of human effort required for manual evaluation methods:

Automatic scoring of test points [10] – this would involve developing a translation, where each segment had a different translation issue. The file is translated and compared against a set of acceptable translations (in Chinese in the original study) and scored based on the translation issue for that particular segment.

Evaluation using n-gram co-occurrence [11] – These methods involve determining how “close” a translation is to a series of expert translations. Of course, the further the translation is from the expert translations, the poorer quality it is seen to be.

Edit distances – this involves measurement of the quantity of edits required to take a machine translation and make it human (or in our case, make it of higher quality). This is where the “number of insertions, deletions and substitutions required to convert one string into another” [5] is measured.

In the context of evaluating human translations, these approaches may be less appropriate. Automatic scoring of test points would involve a time consuming process of creating different translation segments with particular translation issues, as well as obtaining translations,, possibly more than one for each segment. Evaluation using n-gram co-occurrence appears to be most appropriate in situations where there is likely to be a significant difference between the initial translation and the expert translation. In a human translation context, it is likely that the differences will be small and more related to style.

2.3 Usability

Jakob Nielsen defined usability by five quality components [12]:

- Learnability: How easy is it for users to accomplish basic tasks the first time they encounter the design?

- Efficiency: Once users have learned the design, how quickly can they perform tasks?
- Memorability: When users return to the design after a period of not using it, how easily can they reestablish proficiency?
- Errors: How many errors do users make, how severe are these errors, and how easily can they recover from the errors?
- Satisfaction: How pleasant is it to use the design?

All these factors will be important in designing a tool for the localisation industry. Any tools developed must be easy to learn so they can be quickly used in a production environment, have high efficiency to fit in with translators working patterns and methods of working quickly. There is no guarantee that a translator will be translating software on a regular basis so memorability will be a factor. “Errors” will be important, as the translators are working in a high pressure time sensitive environment, and any major problems the translator has with the software could endanger a project completion. Satisfaction is of course a factor as a more pleasant tool to use will be taken up more willingly by translators. The tools will have to be designed with these factors in mind and usability evaluation will be designed to examine them.

3 Field Research

The research methods chosen were semi-structured interviews and some observation of working practices. A mass distributed questionnaire was ruled out because the purpose of the research was to gain rich descriptive information about the problems of software translation and this would be difficult to obtain in a list of questions sent out to a translator. I have a great deal of access to a small group of translators and felt it would be more useful to interview them personally. A focus group style meeting was a possibility also but the translators I have access to have wide ranging personalities and there was a possibility that individuals could dominate proceedings. A skilful facilitator would be required.

Face to face semi-structured interviews were carried out with 5 translators, a TRADOS Specialist and a Software Localisation engineer, with the aim being to identify problems they had with the current software localisation process and if they had any suggestions for improvement to the process and TRADOS. Discussions with the TRADOS specialist also involved discussing how introduction of new tools would affect processing.

The results (see Appendix C for notes of this research) confirmed that translators would be supportive of any system that allowed the visualisation of software files would be of great benefit. The problems they most encountered were problems of having to shorten translations to deal with the size of bounding boxes, and changing translations due to them being incorrect in context with the rest of the software.

Requests were made regarding a more sophisticated verification tool for TagEditor, checking whether segments have been translated. However this is part of an existing development by Xerox and it would be inappropriate to develop anything in this area.

The Software Localisation Engineer interviewed stressed that any means of cutting the time spent on validation would be beneficial, as the engineer must support the validators work. Also, they spend a great deal of time making changes to hotkeys and fixing bounding boxes for truncated strings that the validator could not correct.

The TRADOS specialist interviewed expressed that any tool introduced would have to be such that it didn't significantly slow down pre and post processing. .resx files take a level of manual processing so a tool that automated that work would be helpful.

Observation of a translator showed that with experience translators tend to look for ways to work quickly. This particular translator made a great deal of use of shortcut keys. They also refer often to information packs sent to them by terminologists, explaining more about the project and common problems with it.

4. Preparation of Solution

4.1 Choice of solution

Based on the background and field research , the following solutions were decided upon.

- A tool that integrates into TRADOS which visualises TTX files created from ..resx windows resource files. As a feature of this, bounding boxes will be checked to confirm whether the text has exceeded their bounding limits, and hotkeys will be checked for duplications.
- The comments system of TRADOS will be extended so that translators notes from earlier projects can be referenced in subsequent translations.

The decision to create a visualisation tool was based on the very strong support for it from translators and it has been documented as an issue within software translation (2.1.3). It was established that it had to be available as a plug-in to TRADOS after examining the other software localisation tools on the market (2.1.4) and hearing Xerox and translators' preferences towards that software – TRADOS supports the current Xerox business model which utilises a number of external translators. The bounding box and hotkey checking functions are also known issues in the industry (2.1.3). Extending the comment functionality of TRADOS also received a positive response from technical staff, but was seen as less important than the visualisation tool.

4.2 Methodology

The traditional waterfall model has the following stages: Initiation, Feasibility, Analysis, Design, Build, Implementation/Changeover, Maintenance and Review [13].

Not all of these steps will be applicable in this project. A "Changeover" will not occur in the course of the project itself, because the tool will not be permitted for use on production projects until a thorough evaluation and testing has taken place, and the timescale involved will not allow for that. Since it won't be put into production use during this timescale, maintenance will not be involved. An advantage of the waterfall model is the fact that all requirements are identified at the outset of the process. However, the waterfall model is considered to be rigid and if any problems occur or requirements change, it is thought to be difficult to return to earlier stages.

Rapid applications development (RAD) [14] is a methodology that suggests a much quicker movement through the process from initiation to completion, but will pass through several iterations of this to guarantee quality. RAD projects are believed to work best where the tasks involved are

small and defined, the number of team members is also small, and where each member is versatile enough to work on different parts of the process eg. the analysis as well as the coding. These factors seem to be in line with this project. RAD is often used in conjunction with time boxing, where implementation decisions are based on how much time is available.

An iterative model (where several rounds of requirements gathering, coding and evaluation would be carried out) for developing the tools was ruled out due to time constraints. It would be difficult to obtain enough time to involve translators, TRADOS specialists and localisation engineers perhaps 3 or 4 times in the process of gathering requirements and evaluating the tool. All of these people work on production projects for 90% of their working time and it is difficult to predict when there will be a significant enough down period to allow for research and testing. A waterfall style approach was adopted because it will be simpler to obtain the relevant people for one period of requirements gathering and one period of evaluation. Evaluation after initial testing of the software, will consider both usability and potential improvements in quality.

4.3 Requirements Gathering

The field research performed helped to establish what the requirements of these solutions would be. Staff at Xerox were sent a requirements specification based on what had been understood from the research, together with sample GUI's (Appendix D). A meeting was conducted involving translators, technical staff and management to discuss this specification. The following was decided upon

.resx Visualisation

- A preparation tool that ensures that non-text properties of the .resx file are not available for translation.
- A visualisation of the .resx file in TRADOS TagEditor
- A view of both source and target text.
- Bounding Box checking – If a bounding box is exceeded its text is displayed in red
- Hotkey checking – if hotkeys are duplicated, the text is marked in blue
- Highlighting of the segment of text that is currently open for translation.
- HTML reports generated for both bounding box and hotkey checking

Extending TRADOS comments functionality

- The ability to generate an HTML report from the TTX comment file, displaying the source and target text as well as the comment made.

- A plug-in in TagEditor which “pops up” showing any comments related to the TTX file currently being translated.

4.4 Project Plan

	Start	End
Background Research	-	19/01/07
Field Research	1/12/06	22/12/06
Analysis + Design	28/12/06	19/01/07
Coding Part 1 – Visualisation	22/01/07	16/02/07
Coding Part 2 – Comment Functionality	19/02/07	02/03/07
Testing	05/03/07	16/03/07
Evaluation	19/03/07	30/03/07
Report	31/03/07	13/04/07

Figure 4.1 – Project Plan

With the selection of methodology and the requirements gathering in mind, the remainder of the project plan was developed in greater detail. It had been understood from the interviews and requirements gathering process that the visualisation software was of greater importance, so more time is dedicated to this in the plan.

Background research was planned to be ongoing until January in order to establish potential means of evaluation. The timescale for the coding was decided based on experience of developing other software, with consideration for the requirements specification (4.3). The coding of the preparation tool was included in the “Visualisation” portion of the coding as it is linked to that deliverable.

Testing occurs in a less structured manner throughout the coding phase, but a formal testing phase was planned for after its completion. This was planned to be functional testing and two weeks was assigned to it as an estimate. Evaluation involving end users was planned for a 2 week window, as some flexibility was required to fit into Xerox operations where production work may occur at short notice. Although the report writing was an ongoing process, all project time in the plan was dedicated to it once the evaluation had been carried out. This plan meant that the project would have reached completion 2 weeks prior to the final deadline.

4.5 Selection of Development Language

Since the solution takes advantage of TRADOS' SDK, which consists of .NET dll files, it is necessary to code the solution using one of the .NET languages: C#, J# or VB.NET. These languages share libraries, execution speeds and the editing environment Visual Studio, although J# does not have the advantage of automatically created unit tests, which the others do. J# though does have the advantage of the availability of the Java libraries up to version 1.1.4 .Much of this decision is down to individual tastes. Xerox developers expressed no particular preference, since the developers in the organisation are skilled in a number of languages and would be comfortable adding/editing the software if necessary, regardless of language. The author has a greater knowledge of Java so J#, with its identical syntax and similar libraries, appeared to be the correct choice.

5. Design & Implementation

This chapter discusses the challenges faced when implementing the solutions, and the process by which they have been achieved. This includes a discussion of the TRADOS SDK (the means of connecting to and utilising TRADOS), the file types involved in the implementation, as well as the design and coding of the visualisation and comment functionality. This chapter also documents the functional testing carried out on the software prior to its evaluation.

5.1 Affected file types

3 different file formats are affected by the 2 different tools produced by the solution and a major challenge of this project was to understand the structure of these files to obtain the correct information.

5.1.1 .resx files

.resx are a Windows .NET resource file. It can store, if requested, all coordinate information for UI components, images, icons and locale specific text strings within a user interface. This data is stored in an XML format (figure 5.5)., and from this it was discovered it would be possible to generate a version of the user interface, without having the actual source code files.

Within .resx files, each UI component has a separate <data> tag for its properties which include its type (eg. Button, textbox), its text, size. Each of these data tags has an associated <value> tag to assign that value. These exist as long as the properties are assigned to a value which differs from the default. For instance, if no font has been set, the default is Microsoft Sans Serif, size 8.25pt, and this will not be found in the .resx file.

If changes are made to a .resx file within a Microsoft Visual Studio project, the user interface it refers to will automatically update to reflect those changes. To visualise this content, it is necessary to understand this structure. Once each attribute value has been obtained, it is then possible to visually “draw” a component.

5.1.2 .TTX files

TTX is the bilingual file format used for editing in the TRADOS TagEditor environment. It is xml-like and builds on the format of the source file that requires translation eg. if a .resx file is to be

translated, it will contain all of that file's content plus the TTX file's tagging. This TTX tagging dictates how the file will be displayed in TagEditor. Non-translatable text eg. reserved words, XML tags, etc. is "blocked." This is determined by a Settings file where users can decide whether the contents of a particular tag is translatable. Also, it will either be set as an external of an internal tag. This determines whether it should be included in a translation unit eg. a bold tag in html would likely be an internal tag, as it has a direct effect on the context/understanding of the text. Whilst tags to create a table would be external as it has no bearing on the linguistics of a phrase/sentence. External tags can be moved, but internal tags cannot..

Once a translation has been entered, either automatically by translation memory or by a linguist, that segment is placed within a <TU> tag. This tag contains attributes such as languages of that translation unit and it's percentage match from translation memory.

```
 <ut Type="start" Style="external" RightEdge="angle" DisplayText="data">&lt;data
name=&quot;button7.TabIndex&quot; type=&quot;System.Int32, mscorlib&quot;&gt;</ut>
   <ut Type="start" RightEdge="angle" DisplayText="value">&lt;value&gt;</ut>16<ut
Type="end" LeftEdge="angle" DisplayText="value">&lt;/value&gt;</ut>
 <ut Type="end" Style="external" LeftEdge="angle" DisplayText="data">&lt;/data&gt;</ut>
 <ut Type="start" Style="external" RightEdge="angle" DisplayText="data">&lt;data
name=&quot;button7.Text&quot; xml:space=&quot;preserve&quot;&gt;</ut>
   <ut Type="start" RightEdge="angle" DisplayText="value">&lt;value&gt;</ut>Browse<ut
Type="end" LeftEdge="angle" DisplayText="value">&lt;/value&gt;</ut>
 <ut Type="end" Style="external" LeftEdge="angle" DisplayText="data">&lt;/data&gt;</ut>
 <ut Type="start" Style="external" RightEdge="angle" DisplayText="data">&lt;data
name=&quot;&amp;gt;&amp;gt;button7.Name&quot;
xml:space=&quot;preserve&quot;&gt;</ut>
   <ut Type="start" RightEdge="angle" DisplayText="value">&lt;value&gt;</ut>button7<ut
Type="end" LeftEdge="angle" DisplayText="value">&lt;/value&gt;</ut>
 <ut Type="end" Style="external" LeftEdge="angle" DisplayText="data">&lt;/data&gt;</ut>
```

Figure 5.3 – a sample TTX file as it would be seen as plain text

5.1.3 TTX comment file

When a comment is added to a TTX file in TagEditor, a "Comments" file is generated. This file contains line number and offset information, as well as comments made about particular translation segment(s).

To make this of value to future versions of a project, it would be necessary to link together the comment with the source and target translations in a single file. Currently the comments are only of

use to someone looking at the same TTX file, and would probably mainly be of use for validation purposes

```
<?xml version="1.0" encoding="utf-16"?><File><Comments /><Segments>
<Segment>
<Location>
<StartParagraph>229</StartParagraph>
<StartOffset>11</StartOffset>
<EndParagraph>229</EndParagraph>
<EndOffset>11</EndOffset>
<SegmentReference>teSegmentReferenceSource</SegmentReference><LocationType>teLoc
ationTypeSource</LocationType><FileName>C:\Documents and Settings\User\My
Documents\Visual Studio
2005\Projects\Form1..resx.TTX</FileName></Location><Comments><Comment
severity="Medium" user="User" date="2007-03-01T10:50:37" version="1.0">This isnt the
right product terminology
</Comment></Comments>
</Segment>
</Segments></File>
```

Figure 5.4 – TTX Comment file

5.2 Using the TRADOS SDK

The Trados software development kit is a collection of code libraries (Windows dll files) which can be used to manipulate and automate TRADOS functionality. There are separate libraries for Translators Workbench, TagEditor as well as all the other Trados tools. In the past at Xerox it has been used to automate the creation of memories and to create verification plug-ins in TagEditor, to check whether segments have been translated.

For TagEditor in particular, there are a number of "events" for which event handlers can be added , so that any plug-in can react to them. For instance, if you wished for some code to execute when the user saved a document, you would add an OnSaveEventHandler to your code, which calls a method passed to it.

```
application = new ApplicationClass();
saveHandler = new
_IApplicationEvents_OnAfterSaveBilingualEventHandler(this.OnAfterSaveBil
ingual);
application.add_OnAfterSaveBilingual(saveHandler);

…

public void OnAfterSaveBilingual(TagEditor.Document document)
{

}
```

Figure 5.1 – Code to implement an event reacting to a file being saved in TagEditor

The application object (the currently open TagEditor application) has an onAfterSaveBilingual event added to it, which has a method name as a parameter.

Whilst examining the SDK in detail it was established that it had some severe deficiencies that could adversely affect the solutions delivered by this project. Some events that it was expected would be available as part of the TagEditor libraries were not, most importantly ones relating to the opening and closing of translation segments. This caused particular issues for the visualisation tool (see section 5.4).

5.2.1 Implementing TRADOS Plug ins – COM

Although it is not completely obvious from the TRADOS SDK documentation, to enable software as a TagEditor plugin, it first must be built as COM (Component Object Model) classes. COM is Microsoft technology which “enables software components to communicate. COM is used by developers to create re-usable software components, link components together to build applications, and take advantage of Windows services.” [15] Each class must be registered by inserting assembly information into the header of the classes. By making these classes visible to COM, TagEditor can access them.

The code to implement a COM class is as follows:

```
import System.Reflection.*;
import System.EnterpriseServices.*;

/**@attribute Transaction(TransactionOption.Required)
 *@attribute ProgId(".resxReader..resxRead")
 */

public class ResxRead extends ServicedComponent
```

Figure 5.2 – Code to implement a COM Class

To complete the registering of a TagEditor plugin, some windows registry editing must occur. The class identifier (CLSID) - a unique identifier - must be set to implement a particular category. A category is a group of classes that the application can take advantage of. In this case the category is that of a TagEditor plug-in, so that TagEditor can view the make use of the plug-ins created. This area proved to be a great challenge as the project was embarked upon with no prior knowledge of COM and limited knowledge of Windows registry editing.

5.3 Preparation tool for .resx files

For .resx visualisation to take place, all of the property information eg. Coordinates, must be included in the .resx file. However, these properties are represented in the .resx file in the same way as xml tags containing the text for translation (figure 5.5):

```
<data name="label1.Size" type="System.Drawing.Size, System.Drawing">
  <value>41, 13</value>
</data>
<data name="label1.TabIndex" type="System.Int32, mscorlib">
  <value>13</value>
</data>
<data name="label1.Text" xml:space="preserve">
  <value>Source</value>
</data>
```

Figure 5.5 – .resx File

Translatable text within .resx files is determined by TRADOS as any text that exists within <VALUE> tags. This could be a major pitfall, as any manipulation of this text by linguists could result in rebuild/compilation problems when these files are used to construct the completed software.

To handle this issue, a preparation tool was developed. This work would normally be carried out by technical staff prior to translation, but this application automates the process.

Figure 5.6 shows the user interface for the preparation tool. The user supplies a list of .resx files and the location the prepared files should be copied to.. The program takes a copy of each file and reads through its contents checking each line with a regular expression. The expression searches for data tags with a pattern ‘name=”*.*” ‘. If it locates this pattern, it checks whether the second wildcard is the word “Text”. If not, the next line is read, and the value tag associated with it has its name changed. This means that this property information can be made unavailable before translation, as it now has a different structure to that of the text sections. After translation, the process can be reversed by selecting the “Post Translation” option.

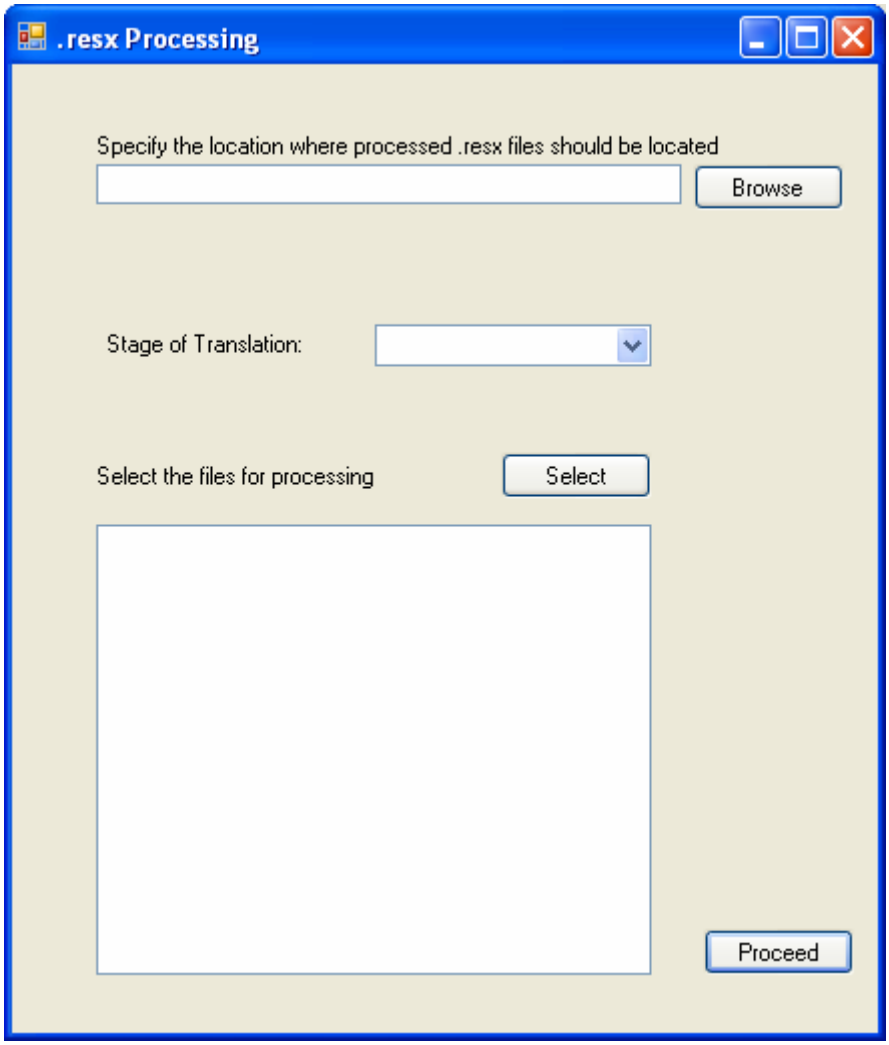


Figure 5.6 – GUI for .resx Preparation tool

5.4 Visualisation of .resx TTX files

5.4.1 Class design

The classes designed in this deliverable, each have their own particular purpose rather than representing an entity. The .resxReader class reads and displays the contents of the TTX file. The BoundingBoxChecker class takes a UI component and checks whether the text has exceeded it's bounding box restrictions. The HotKeychecker class takes a list of UI components and checks whether the hotkeys specified for each component have been duplicated. These are all called from the "Visualisation" class that establishes a connection to TRADOS and reacts to TRADOS events. The functionality was separated into these classes because in the event of a change in structure of the other classes, the others would require little or no changes.

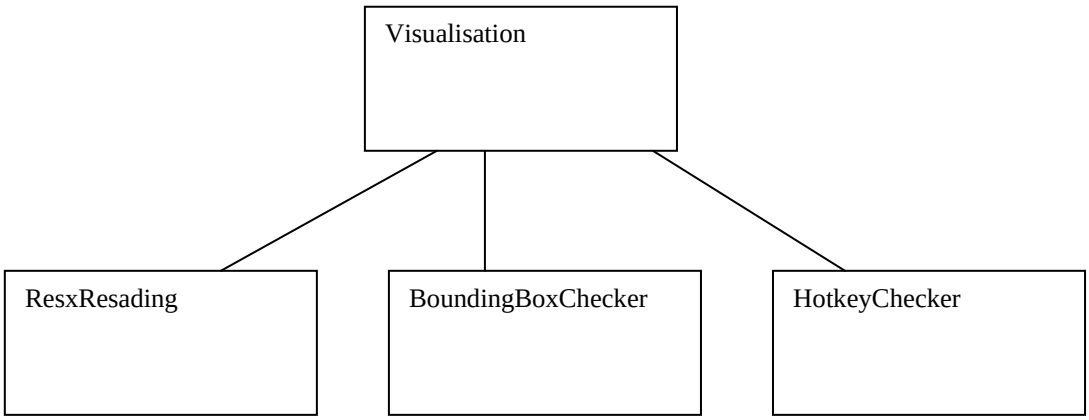


Figure 5.7 – Class Diagram for .resx visualisation tool

5.4.2 Regular expressions

Although regular expressions are used in some way in each of the deliverables, they have greatest impact in the Visualisation tool. To obtain the required information from the TTX file, it is necessary to create a number of regular expressions - patterns of text used for searching. It was necessary to learn the format of the Microsoft .NET style regular expressions which are used in the libraries of J#

and the other .NET languages. For instance figure 5.8 below shows an expression which finds the width and height of a UI control.

```
Size size = new Size();
int width = System.Convert.ToInt32(Regex.Match(line2,
"value&gt;\\</ut\\>\\<ut Style=\"external\"
DisplayText=\".+\"\\>(?<x>(.+)),
(?<y>(.+))\\</ut\\>\\<ut").get_Groups().get_Item("x").get_Value());
int height = System.Convert.ToInt32(Regex.Match(line2,
"value&gt;\\</ut\\>\\<ut Style=\"external\"
DisplayText=\".+\"\\>(?<x>(.+)),
(?<y>(.+))\\</ut\\>\\<ut").get_Groups().get_Item("y").get_Value());
size.set_Height(height);
size.set_Width(width);
buttons[count].set_Size(size);
targetButtons[count].set_Size(size);
```

Figure 5.8 – Example Regular expression code used to obtain UI characteristics from TTX files

It sets up 2 wildcards as variables, width and height, so that they can be extracted, converted from a String to an integer and used with the UI controls set_size() method.

5.4.3 Approach to reading .resx.TTX files

The process used for reading the UI data from the TTX file is as follows:

- The class is supplied with a TTX file, which is opened for reading.
- The characteristics of the main form are located to determine the size of the visualisation window.
- A TabControl and two TabPages are added to the form. These are entitled “Source” and “Target”.
- Regular expressions are used to find any <data> values which fit the pattern ‘*.type’, where * is a text string.
- Its <value> data, found on the next line, is read to identify the component type eg. Button, TextBox
- Two components are created in two separate arrays of the identified component type. For example, if a Button is located, Buttons will be added to a source array of buttons, and the other to a target array. One will be set to appear on the source tab, and the other will appear on the target tab.
- Regular expression searches are used to establish the location, height, width, text and parent of the component. These properties are set within the previously created control.

- If a translation unit exists, the text for the source control will be taken from the source tag, and the translated control from the target tag. If not, both components will have the original source text.
- The control is set to visible
- The whole form is made visible.

All UI components are then stored in an ArrayList. Those UI components which could potentially have hotkeys associated with them are also stored in another list for use by the HotKeyChecker class (5.4.6).

5.4.4 Optimising the TTX reading code

All components of a GUI have largely similar properties that must be “read” into the visualisation eg. Size or location, but some have their own particular characteristics. For example, a TabPage will have a value indicating its position relative to other TabPages.

To avoid code repetition, separate methods have been developed to a) create each component and to establish its unique properties (in a method newX() where X is the name of the component) and b) locate and set all common properties (in a method newComponent()). However it was discovered that some components did not inherit their methods from the type “Control”, which is the case for the majority. These components needed to have the information for them searched for separately eg. For ToolStripMenuItem’s, all properties were set in its newMenuItem() method.

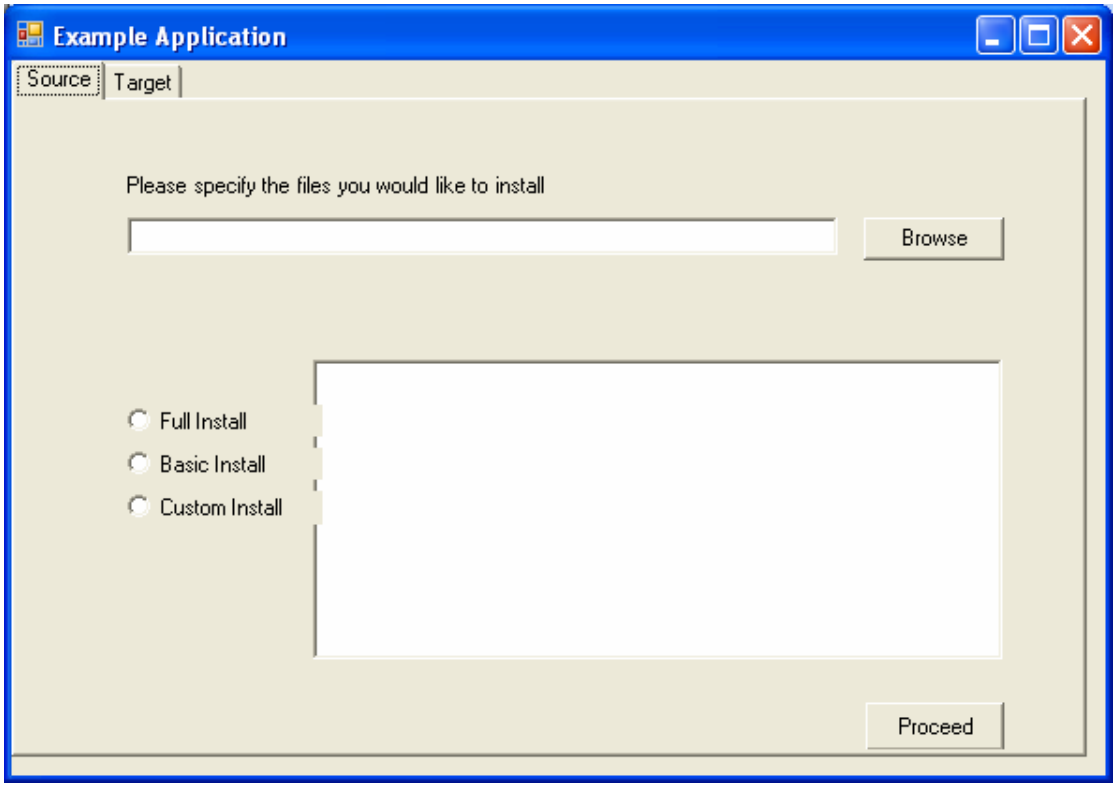


Figure 5.9 – Example of source visualisation view

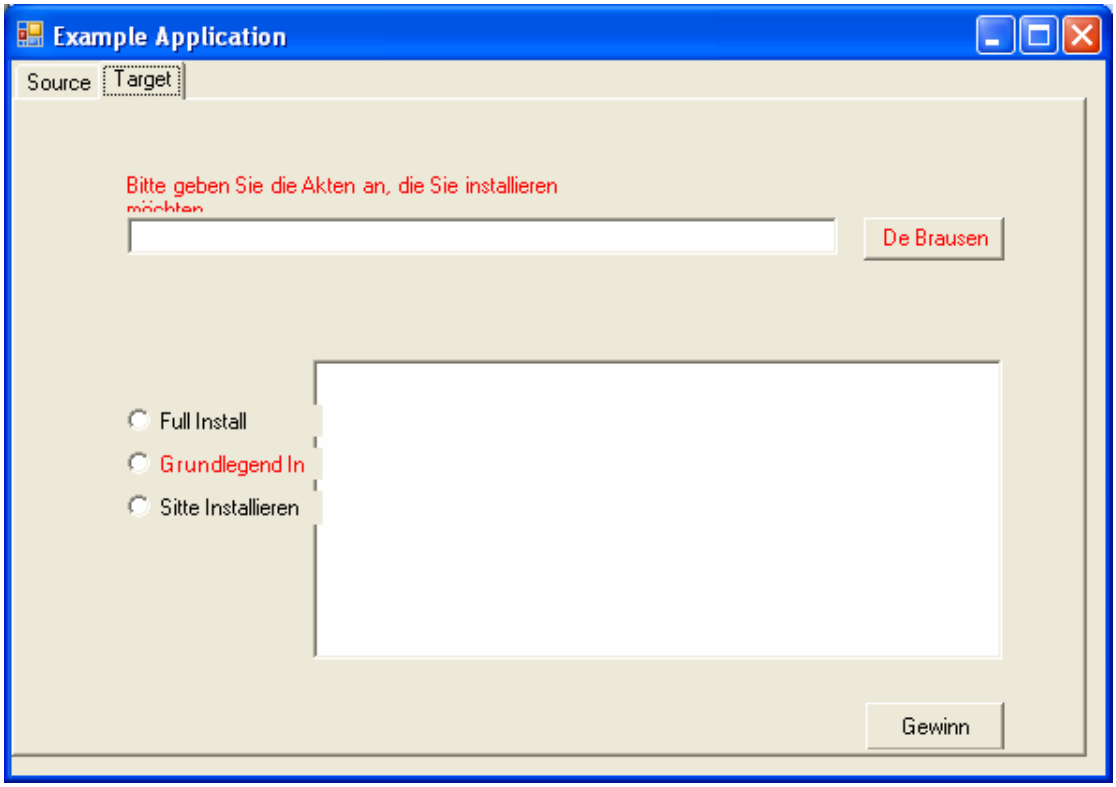


Figure 5.10 – Example of target visualisation view

5.4.5 Approach to measuring bounding boxes

The class “BoundingBoxChecker” class is passed a UI component, and obtains its text, font and size using the object’s various “get” methods. Using a .NET library function (TextRenderer), the width and height of the text in pixels is measured taking the font type and size into account. It then compares this figure against the UI components specified size, and if this is exceeded, the UI component will be returned with its text marked in Red. It is also added to a list of components that have exceeded their boundings.

In practice this is called from the main program class, which will take the ArrayList of source UI components, and loop through all of them checking whether they have exceeded their bounding box.

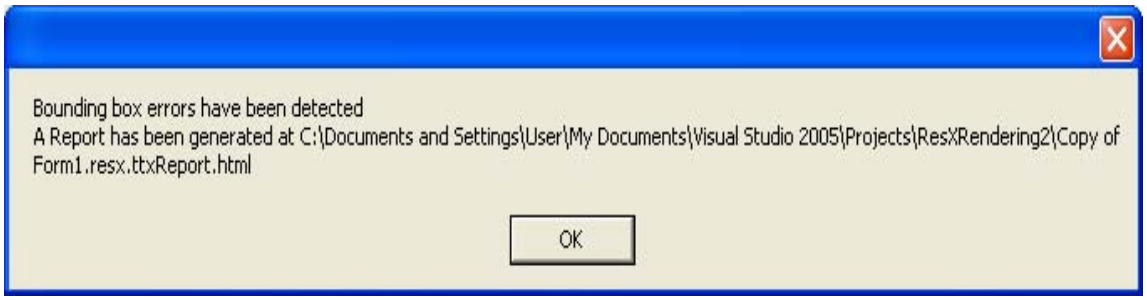

Bounding box errors have been detected
A Report has been generated at C:\Documents and Settings\User\My Documents\Visual Studio 2005\Projects\ResXRendering2\Copy of Form1.resx.ttxReport.html
OK

Figure 5.11 – Example bounding box error message

On completion, an HTML report is generated which displays the source and target text, the size of the current translated text in pixels, and the bounding box size (Figure 5.12). It does this by looping through the list of exceeded controls, getting the source and target text as well as size information. using the components “get” methods. Since this is an HTML file rather than the Unicode encoded TTX or .resx file, the text supplied to the report must be HTML encoded eg. ä replaced with ä, but will be displayed as the character. This is especially important in a translation environment as there will be far more special characters being used. After some searching it was established that a library method called HTMLEncode could be used for this. This method appears to have been designed for use with ASP.NET and is not normally available for use in a Windows Form application, so it was necessary to add it as a reference dll. This has been applied to the Hotkey (5.4.6) and comment reports (5.5.1) as well.

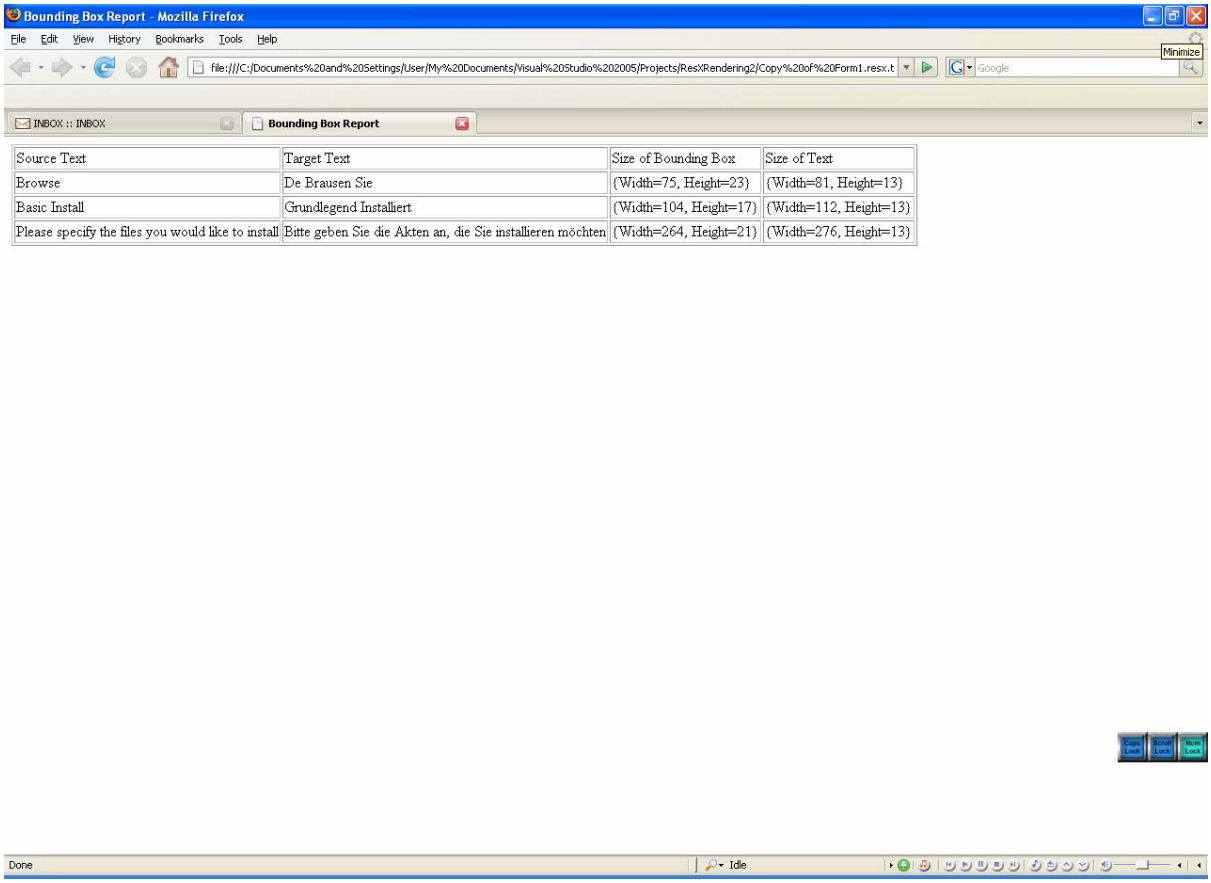

Bounding Box Report - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Minimize
INBOX :: INBOX
Bounding Box Report
Source Text | Target Text | Size of Bounding Box | Size of Text
Browse | De Brausen Sie | (Width=75, Height=23) | (Width=81, Height=13)
Basic Install | Grundlegend Installiert | (Width=104, Height=17) | (Width=112, Height=13)
Please specify the files you would like to install | Bitte geben Sie die Akten an, die Sie installieren möchten | (Width=264, Height=21) | (Width=276, Height=13)
Done
Idle

Figure 5.12 – HTML bounding box error report

5.4.6 Hotkey checking

The HotkeyChecker class works in a similar manner to that of the BoundingBoxChecker. It is supplied an ArrayList of UI components, but in this case they are exclusively ToolStripMenuItem as opposed to all of the UI components in the bounding box checking. The hotkeys for all of these items are stored in a String array, and these Strings are all compared to each other. If a hotkey duplication is detected, the UI component is added to a list of duplicates and its text is set to blue in the visualisation window.

The HTML report generation is also similar. The header information of the HTML file is written, and then the list of UI components with duplicated hotkeys is looped through, adding each ones text and hotkey to the file.

5.4.7 The main visualisation program

When the visualisation tool is activated, the main visualisation class acts as a boundary, calling the ‘reading’ and ‘checking’ classes. It searches for an existing TagEditor object ie. the open application, and gets the currently active document. A copy of that file is made, and subsequently supplied to the

ResxReader object for ‘reading’. Once completed, the bounding box and hotkey checking classes are called. The bounding-box checker class is supplied with every target UI component to check.

This main class also has a number of event handlers. Initially it was intended that the segment currently being translated would be highlighted in the visualisation window. However due to the limitations of the TRADOS SDK, this was not possible as there were no events available to identify what translation segment was open and when it was opened or closed. As a result of this, another means of refreshing the view had to be implemented. When a file is saved in TagEditor, the visualisation window refreshes to show any changes made by the linguist, and re-run the bounding box and hotkey checkers. When TagEditor is closed, the visualisation window will also close.

With Nielsens “efficiency” usability principle in mind [17], a shortcut key command was built in so that the visualisation window would refresh if the user pressed Shift+r. This was to help the translator use the software quicker if they chose to.

5.5 Extending TRADOS Comments functionality

The design of this functionality is in two distinct parts; one to generate a comment report and another to act as a Trados plug-in to display relevant comments to the currently open TTX file.

5.5.1 Generating a comment report

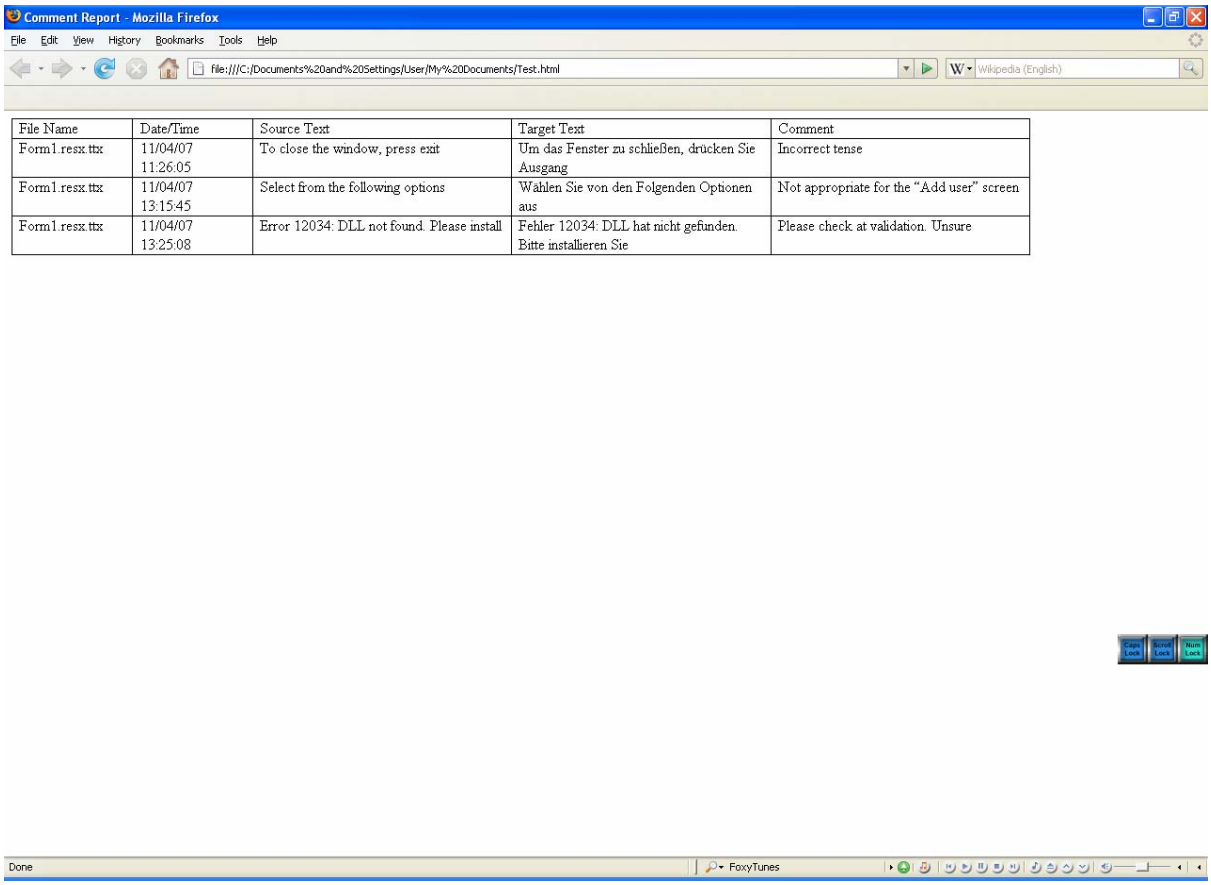
The comment report is generated by selecting it from the context menu for a .TTX.comments file in Windows explorer . This is achieved by editing the registry – the selected filename is supplied to the program as an argument.

Comments are generated by using the source English and TTX file, and the comment files in the following process

- Identify a comment in the comment file
- Take the line and offset information from it, and identify the correct line in the TTX file.
- Hold this text in a variable.
- Search for line and offset in the TTX file , to locate the source and translated text the comment relates to.
- Add the source and translated text, along with its comment to a report.

In identifying the most appropriate way of gathering information from the comment file, some time was spent searching the Microsoft Developers Network (MSDN) libraries [16] to find out whether libraries existed specifically for looking through XML files. The XMLTextReader class allows the user to identify particular tags in XML without using more cumbersome regular expressions, which are appropriate in the visualisation portion of this project, where the input is more complicated but in the case of the comment file and looking for translation units in a TTX file, these can be found more simply.

Figure 5.13 shows how the report is formatted.



File Name	Date/Time	Source Text	Target Text	Comment
Form1.resx.ttx	11/04/07 11:26:05	To close the window, press exit	Um das Fenster zu schließen, drücken Sie Ausgang	Incorrect tense
Form1.resx.ttx	11/04/07 13:15:45	Select from the following options	Wählen Sie von den Folgenden Optionen aus	Not appropriate for the "Add user" screen
Form1.resx.ttx	11/04/07 13:25:08	Error 12034: DLL not found. Please install	Fehler 12034: DLL hat nicht gefunden. Bitte installieren Sie	Please check at validation. Unsure

Figure 5.13 – Comment Report

5.5.2 Displaying comment information to translators on future projects

This deliverable was designed in a manner consistent with that encouraged by proponents of UML [17]. It has a boundary class, the "TTXCommentViewerUI" class, which deals with interaction

between the system and its user, a control class, “TTXCommentViewerControl”, which handles the reading of comment an TTX files, and an entity class “Comment”, which holds the information about a particular comment. This is done so that “any changes to the interface or communication aspects of the system can be isolated from those parts of the system that provide the information storage or business logic” [17].

When there are comments related to translations that could be used in the current TTX file, a pop-up window will appear alongside TagEditor to show these comments. On the opening of a file in TagEditor, the program will obtain a Document Object (the TTX file). The existing comment report(s) are read through, the TTX file is read (using the XMLTextReader library class used for the report generation) checking whether the commented segment is found in it. If so, a window is initialised containing the source text, its translation and the comment about it (screenshot in figure 5.14).

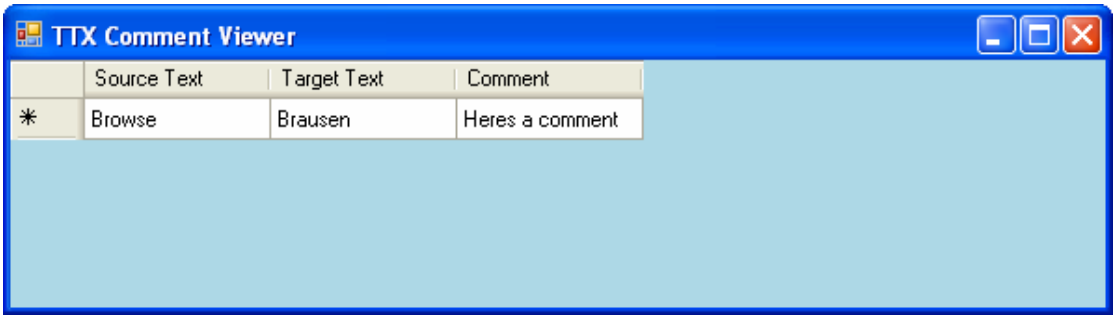


Figure 5.14 – TTX Comment Viewer

5.6 Documentation

To accompany the software deliverables, there are two documentation deliverables – a User guide and a Developers guide.

The Developers guide documents the class structure of both tools and explains the purpose of each method. (Appendix E). The purpose of this is to enable developers at Xerox to update the software or fix bugs should they emerge at a later date.

The User guide documents the installation process, how to activate the plug-ins within TagEditor and how to use the software. Although it is an expected accompaniment to software, it enhances its usability by making the user more aware of its basic functionality and accelerating its use, for example the use of shortcut keys. In both cases clear documentation is necessary to limit the need for training. Translators are often geographically distant and on site training is impractical.

5.7 Testing

A comprehensive test plan has been formulated to ensure that the software performs all of the tasks that were initially scoped for this project (Section 4.2). Although elements of these tests were iteratively used during development, it was necessary to have a more structured test towards the end of the implementation, so that it could be checked that all functionality integrated together correctly and behaved as expected.

The form of testing applicable here, Functional testing, known as ‘Black box’ testing, is based upon the requirement specifications of the system and tries to map the input values to the output values. The system may also be tested with some invalid inputs to judge its behaviour and find out its exception handling ability. In order to test the functional correctness of the features, test cases can be generated [18]. This technique was chosen as it represented a means of testing against a number of typical user scenarios.

5.7.1 .resx Visualisation

The tool was supplied with a variety of different .resx/TTX files, all representing a different user interface, some of which were supplied by Xerox. These were used to determine if the GUIs were displayed as they had been originally intended. Translations were added and modified to confirm that the window refreshed correctly. Ie. Correct dimensions, text, font.

Bounding-box checking was tested using .resx files containing a variety of different fonts and font sizes. Each font has different pixel sizes which would affect the UI visualisation and whether or not it had exceeded bounding limitations. The following questions were asked to confirm the bounding box checking was working successfully:

- Are errors displayed in red?
- After they are corrected do they appear in their normal colour?
- Are all errors included in the report?

File	Visualised Correctly	Correct Bounding Box Checking	Correct Hotkey checking	Correct Refreshing
1				
2				
3				
4				

Figure 5.15 – Sample .resx visualisation test table

Also, how it handles extreme cases will be tested ie. what if the translation is extremely long or there is no translation at all?

5.7.2 Results

The first run through this test plan produced the following results

File	Visualised Correctly	Correct Bounding Box Checking	Correct Hotkey checking	Correct Refreshing
1	Y	Y	Y	Y
2	N	Y	Y	Y
3	N	Y	Y	Y
4	Y	Y	Y	Y

Figure 5.16 – First run through .resx visualisation testing

As it shows some files were not visualised correctly. In each of these cases it was because a UI component did not display. In one case it was because the code to display it had not been implemented, and on another it was due to incorrect code. These issues were corrected, further tests were carried out successfully (Results found in appendix F)

5.7.3 Comment Functionality

To test the extended Trados comment functionality, the following tests occur:

- Add a series of comments to a file, generate the report and confirm that all comments are included in the report.
- Prepare a new file which should return some of the comments from the report and confirm that all relevant comments in the report files are displayed.

TTX File	All comments included in report	Comments are displayed correctly
1	Y	Y
2	Y	Y
3	Y	Y

Figure 5.17 – Comment Testing table

6. Evaluation

This section looks to evaluate all aspects of the process: evaluation against the minimum requirements, the full requirements specification, the project management and end user evaluation to determine whether the project has achieved its overall goal of improving the software translation.

6.1 Evaluation against requirements

As a tool to visualise the contents of a .resx file in TRADOS was created, the initial minimum requirement was achieved, and the other functionality developed meant that it was exceeded. The visualisation tool correctly presents all of the most commonly used UI components. However it is unable to handle some lesser known/used components or custom ones created by a user.

From the requirements specification discussed with Xerox (Section 4.3), all but one of the requirements was achieved. The ability to highlight the currently translated segment in the visualisation window was not completed due to the limitations of the TRADOS SDK which meant that it was not possible in the timescale. This had a knock on effect that the visualisation could not react to certain options in TRADOS such as “Translate to Fuzzy.”

Compared to the other software localisation tools discussed in this report, the visualisation tool may have less functionality (you cannot make changes to the design of the GUI for instance), but this is not part of the role of a translator and was not considered important to the project. The aim was to integrate some of this software localisation functionality into TRADOS – which is not easily possible with Catalyst and other tools – and this has been achieved.

6.2 Evaluation of Project Management

Largely the project management was effective. The plan (Section 4.4) was adhered to on the whole, but due to the problems encountered in the coding phase, examining the TRADOS SDK and attempting to deal with its shortcomings, the coding overran by a week. Some contingency had been built into the project plan so this was not a catastrophic issue.

The methodology used was appropriate. Although an iterative model may have identified the issues with the TRADOS SDK earlier because of an earlier development stage, the accessibility of Xerox staff meant that a version of the waterfall model was the most sensible option. The requirements were identified in full early in the process.

The selection of J# as the development language posed a number of problems. The implementation of COM classes is automated in VB.Net and C# but not in J#, which led to further coding time. However, the benefits of not having to learn and understand a language with new syntax outweigh this.

6.3 End User Evaluation

There are two components to the evaluation of the software produced. First is to evaluate the effect of the tool on the quality of software translation, primarily in terms of time taken to translate and validate it. Secondly the usability of the software will be evaluated.

6.3.1 Planning

6.3.1.1 Evaluating translation quality improvements

The automatic evaluation methods used in machine translation, described in 2.2 have been ruled out as they appeared to be inappropriate in a purely human translation context. All translation in this environment is carried out by translators of a reasonably high standard, so the initial translation should already be of good quality, and changes made as a result of using the software will mainly relate to terminology and style..

Due to the fact that measuring translation quality is open to a great deal of subjectivity (Section 2.2), it would be difficult to have an evaluator decide whether translations carried out with the software were better than ones performed without it.

The selected evaluation approach, was inspired by the operational evaluation method often used in machine translation evaluation (Section 2.2.2.1), which looks to evaluate cost effectiveness. The translation process used by Xerox makes a time measurement appropriate. A shorter time spent translating software would lead to reduced costs.

Xerox GKLS maintain timesheet data on all translation projects with categories to help future planning eg. X spent 5 hours on the translation and Y spent an hour validating their translation. This will be used as a starting point to analyse improvements.

The process was planned as follows:

- By examining wordcount data from previously translated .resx file projects, identify a translation of approximately 1000 new words (the time metric Xerox use for software translation suggests this should take around half a day to translate and an hour to validate).

The new word figure is based on a weighted calculation where words with no match in the translation memory are valued as one new word and those with fuzzy matches are fractions of a word.

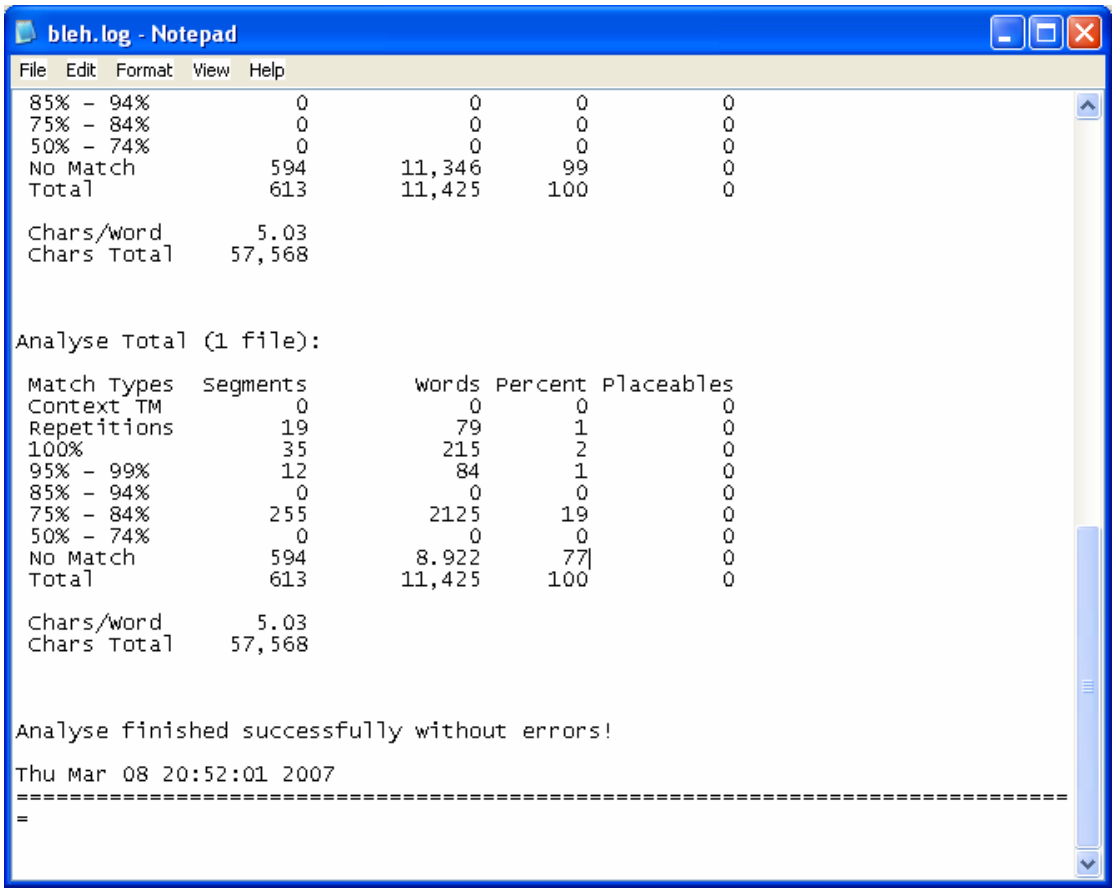

bleh.log - Notepad
File Edit Format View Help
85% - 94% 0 0 0 0
75% - 84% 0 0 0 0
50% - 74% 0 0 0 0
No Match 594 11,346 99 0
Total 613 11,425 100 0
Chars/Word 5.03
Chars Total 57,568
Analyse Total (1 file):
Match Types Segments Words Percent Placeables
Context TM 0 0 0 0
Repetitions 19 79 1 0
100% 35 215 2 0
95% - 99% 12 84 1 0
85% - 94% 0 0 0 0
75% - 84% 255 2125 19 0
50% - 74% 0 0 0 0
No Match 594 8.922 77 0
Total 613 11,425 100 0
Chars/Word 5.03
Chars Total 57,568
Analyse finished successfully without errors!
Thu Mar 08 20:52:01 2007

Figure 6.1 – Example Wordcount Data

- Examine the timesheet data for this project to identify the time spent on translation and validation of this project when initially carried out, and who carried them out.
- With the assistance of a resourcing specialist, who deals with testing and hiring translators, select translators of a similar level to those used in the initial translation.
- Create the translation memory as it was at the time of translation and pre-translate the files.
- Perform the translation, as in any normal project and measure the time taken
- Make any corrections highlighted by the checking functions. Translators were asked to do this at the conclusion of the translation as it would not have been part of the work of the translator initially doing the work.

A reduction in time for translation could suggest less ambiguity over possible translations. Increases in time taken for either task could imply that the tool is adversely affecting the translators ability to carry out their task.

There are a number of potential constraints on the accuracy of the results of this evaluation, and the plan attempts to take these into account. Using a similar quality of translators is important. The translators used at the time of the initial translation may not be appropriate because they could recall the issues they had with the translation and how to deal with them. They may have also improved their translation skills since. All of these issues could affect the time they take to translate the task.

The translation memory must be held at the same state it was at the time of the initial translation. If it is not, the pre translated version of the TTX file will contain a different amount of material to that originally held, and the translator would have a different amount of material to refer to in the TM. Assuming that standard procedures have been followed this should not be a major issue. TTX files for each version of a product are held, usually to check for issues with translations, but in this case the translation memory can be re-prepared to where it was at the time of the initial translation, although it will be somewhat time consuming.

Due to the need for the translators to fulfil production demands it was necessary to make this evaluation quite short, compared to the average size of a software translation that they normally carry out. It will contain approximately 300 new words. Projects of this size are at the bottom end in terms of size of what a normal project would normally be. The translation was to be located by examining earlier wordcount data, which is held centrally by Xerox. The evaluation is also reliant on the timesheet data provided by the translator being accurate.

6.3.1.2 Usability evaluation

If the software created fulfils all of the relevant usability criteria, it will have a knock on effect on translation quality. There are a number of potential methods for evaluating usability which could be used.

Focus groups would involve gathering together those who had used the software created to discuss their feelings on it. However it is thought that focus groups will not provide quality feedback on

problems or how the users actually work with the software [19]. Also, a skilful facilitator is required to organise a focus group out successfully.

Think Aloud evaluation involves the user articulating what they are doing with the software as they use it. The process is videotaped. This would be inappropriate for this particular evaluation because the software performs small defined tasks with limited ambiguity, so there should be very little thought process for the user. Also this process can feel unnatural and perhaps embarrassing for the user [20].

Expert evaluation involves employing a usability expert to use the system and run through scenarios to examine whether this system fulfils usability criteria. This does not seem appropriate to this case because the purpose of the tools is very much focused on localization and having knowledge of this area would be important to understanding their benefits. Also, there is no real cash budget for this evaluation so it would be difficult to obtain a usability expert.

It would seem appropriate to perform a further observation of one or more users, as was performed when understanding the problem (section 3.1). This gives an opportunity to see whether the software fits into the translators working practices.

After the translators have been involved in the evaluation as discussed in 6.1.1, brief interviews are performed to assess the usability of the software. The questions asked focus on if they have found that it benefits their work, and whether it integrates into their working practices. The questions asked look to see whether the software has adequately satisfied Nielsens usability criteria [12] – in this case it will be difficult to evaluate memorability in such a brief evaluation, and lack of errors is mainly covered in the testing phase Although these are semi-structured interviews and could have veered onto other questions, the base questions are as follows:

- Does this software fit into your work well?
- In its current state, would you be happy to continue using it?
- Was it easy to pick up how to use the software?
- Do you have any suggestions for improvements?

It would be difficult to put together a reasonable scenario to show the comments system working as it would in a production situation eg. Looking at comments produced from a previous version of the project. Therefore it was decided to simply demonstrate the comment report generation and comment

viewing functionality to translators and request their thoughts on whether they believed it would benefit their work, and if they had any suggestions for improving it.

6.3.2 Results

The results of the translation improvements evaluation are represented in figure 6.2:
Original time for translation: approx 75 minutes
NOTE – the file translated did not contain any hotkeys

Translator	Time for actual translation	Time for corrections
1	60	5
2	55	10
3	75	5
4	60	5

Figure 6.2 : Translation quality evaluation results

Notes on interviews and observation found in Appendix G.

On the whole the results of the translation quality evaluation would suggest that the tool had been of benefit to translators. All but one of the translations took less time to translate the software than the original translator, and the other carried it out in a similar time to them.

In a normal translation process the translator would not be expected to pay close attention to issues like bounding boxes and hotkeys as they would have little idea as to whether they had exceeded a box or duplicated a hotkey. Having this information at there disposal could have led them spending more time correcting their translations. This would normally be dealt with at validation where an experienced translator could see the built software. Even though they had this information available to them, the translators performed the entire process (translation and fixes) in a similar or shorter time.

The translators were broadly positive about the software when interviewed. They felt that it benefited their work being able to see the built software and being aware of the length restrictions that were placed upon them. They felt it fitted into their working environment well and was easy to learn, satisfying the “Learnability” and “Satisfaction” components of Nielsens usabilty principles [12]. They felt, as was expected, that if the visualisation screen could highlight the segment currently being translated, and refresh on the closing of a segment, then the tool would be improved.

Regarding the comments system, a translator suggested that a “severity" value would be useful, as well as being able to classify different types of comment eg. those relating to terminology, or to do

with grammatical errors. They felt it was difficult to evaluate it's effectiveness in such a short time – they felt the tool would only really show benefits on large projects with a long time gap between updates.

Some feedback was received from technical staff regarding the developers documentation. They asked if more detail could be provided. There was no time available to implement these changes during the project but they will be added after completion.

6.4 Plan for future evaluation

The evaluation carried out here will only give an indication of whether the software produced has improved the software translation process. To gain a full picture the time taken for translation projects will have to be measured over a longer period. Times taken for .resx projects should be recorded and averaged against the new words for translation in that project. This should show whether the time taken on projects is falling beneath the time metric in place at Xerox. A fall would suggest an improvement, in the ways suggested in 5.1.1.

6.5 Conclusions

Based on the various evaluations of the project, it could broadly be considered to be successful. The project has achieved its minimum requirements and the vast majority of its requirement specification, and this occurred in a fashion reasonably close to that initially planned. The deliverables were well received by translators, although they had some suggestions for improvements, and the evaluation of improvements in translation quality produced positive results. Based on these factors, it could be reasonable to suggest that this project has achieved its aim of improving the software translation process.

Discussions in meetings with translators, technical staff and management led to a decision that the software deliverables will be tested further in-house to confirm that they are satisfied that they will be appropriate for use in a production situation when it occurs (there are no .resx translation projects in the near future). They were pleased with the deliverables and were of the belief that they would be of benefit to the organisation.

7. Future Developments

There are a number of potential routes that could be taken to extend the work carried out during this project

7.1 Extending the visualization plug-in

Based on the comments of translators during evaluation, being able to react to segment opening and closing events would have usability benefits. This would involve highlighting the segment to be translated in the visualization window, and refreshing the window on the adding of a translation and closing that segment. Better integration with the translation memory could provide benefits – as the translator cycles through the various fuzzy matches in memory, these could be displayed in context in the visualization window. The segment level interaction was considered for this project but investigation and discussion with Xerox developers suggested that this would be impossible in the timescale. The TRADOS SDK does not provide events for segment opening and closing so it makes this interaction much more difficult. It would involve work to “backwards engineer” TagEditor, as well as some particularly complex work using lower level Windows API’s. This potentially could take six weeks of work alone.

.resx files are not the only type of resource file used in software development. Other formats such as .rc (for Visual C++) and .properties (for Java) are common and encounter the same issues when translated using TRADOS.. The visualization plug-in could be extended to handle these different file formats. These files have less standardized structures than .resx so greater work to interpret them would likely be necessary.

7.2 Extending the comments system

There are two areas of possible improvement/extension of the comment functionality. The first is being able to apply more detail or meaning to the comments provided. This could be accomplished by allowing the user to classify the types of comment they are making eg. Terminological, translation errors. The comments could also be classified by their severity. The system could allow the user to select which type of comments they are interested in seeing.

Also, some investigation could be put into integrating the comments system into translation memory, so that when a segment is viewed in Translators Workbench, the comments related to it are made

visible. Like the segment-level issues with the visualization tool, this could be particularly challenging due to the limitations of the TRADOS SDK.

8. Bibliography

[1] Esselink, B. 2000. *A Practical Guide to Localization*. Amsterdam/Philadelphia: John Benjamins.

[2] Somers, H. 2003. Translation Memory Systems *In*: Somers, H. (ed*) Computers and Translation: A Translators Guide* Amsterdam/Philadelphia: John Benjamins.

[3] *Glossary of terms related to eContent Localisation* .[online]. [Accessed 1st December 2006]. Available from the World Wide Web: http://ecolore.leeds.ac.uk/xml/materials/overview/glossary.xml?lang=en

[4] *SDL TRADOS 2007 Overview.* [online]. [Accessed 10th November 2006]. Available from the world wide web: http://www.lspzone.com/en/products/sdltrados2007/

[5] Bass, J. 2006. Quality in the real world. In: Dunne, K. J. (ed). *Perspectives on Localisation*. Amsterdam/Philadelphia: John Benjamins.

[6] *Alchemy Catalyst 7.* [online]. [Accessed 4th November 2006]. Available from the world wide web: http://www.alchemysoftware.ie/products/catalyst.html

[7] *Passolo Software Localisation Tool | Feature Overview.* [online]. [Accessed 4th November 2006]. Available from the world wide web: http://www.passolo.com/en/features.htm

[8] *Windows Forms Resource Editor (Winres.exe).* [online]. [Accessed 13th December 2006]. Available from the world wide web: http://msdn2.microsoft.com/en-us/library/8bxdx003(VS.80).aspx

[9] Elliot, D. 2006*. Unpublished PHD Thesis.*

[10] Shiwen, Y. 1993. Automatic evaluation of output quality for Machine Translation systems In: *Machine Translation* – Volume 8.

[11] Papineni, K., Roukos, S., Ward, T., Wei-Jing, Z . 2001. *Bleu: a method for automatic evaluation of machine translation* In: *IBM Research Report*, RC22176.

[12] Nielsen, J.1993. *Usability Engineering*. Boston ; London : Academic Press.

[13] Bocij, P., Chaffey, D., Greasley, A., Hickie, S. 2003. *Business Information Systems: Technology, Development and Management for the e-business*. Harlow: Pearson Education

[14] Martin, J. 1991 *Rapid Application Development*. Macmillan Coll Div.

[15] COM: Component Object Model Technologies. [online]. [Accessed 15th February 2007]. Available from the world wide web: http://www.microsoft.com/com/default.mspx

[16] *MSDN Library*. [online] Available from the World Wide Web: http://msdn2.microsoft.com/en-us/library/default.aspx

[17] Bennett, S., McRobb, S., Farmer, R. 2002. *Object-Oriented Systems Analysis and Design Using UML*. .Berkshire: McGraw Hill.

[18] Bader, A. 1997. *Functional Testing*, [Online], [Accessed 15th February 2007] 1st ed, Australia, Monash University, available from the world wide web: http://yoyo.cc.monash.edu.au/~adnan/thesis/paper1.html

[19] *Focus Groups - Usability Methods| Usability.gov*. [Online] [Accessed 24th February 2007].Available from the World Wide Web: http://www.usability.gov/methods/focusgroup.html

[20] Preece, J. Rogers, Y., Sharp, H.2002. *Interaction Design: Beyond Human-Computer Interaction*. New York ; Chichester : Wiley.

The Document Company Xerox. 2006. *Internal Communications*

Appendix A – Personal Reflections

On the whole I have been very satisfied with how this project has been carried out. It has been a very challenging piece of work.

I believe more thorough examination of the tools I needed to work with at the beginning of the project would have shown up problems earlier than they were encountered. If I had been aware of the limitations of the TRADOS SDK in the research phase of the project, the segment interaction requirements in the requirements specification would not have been included.

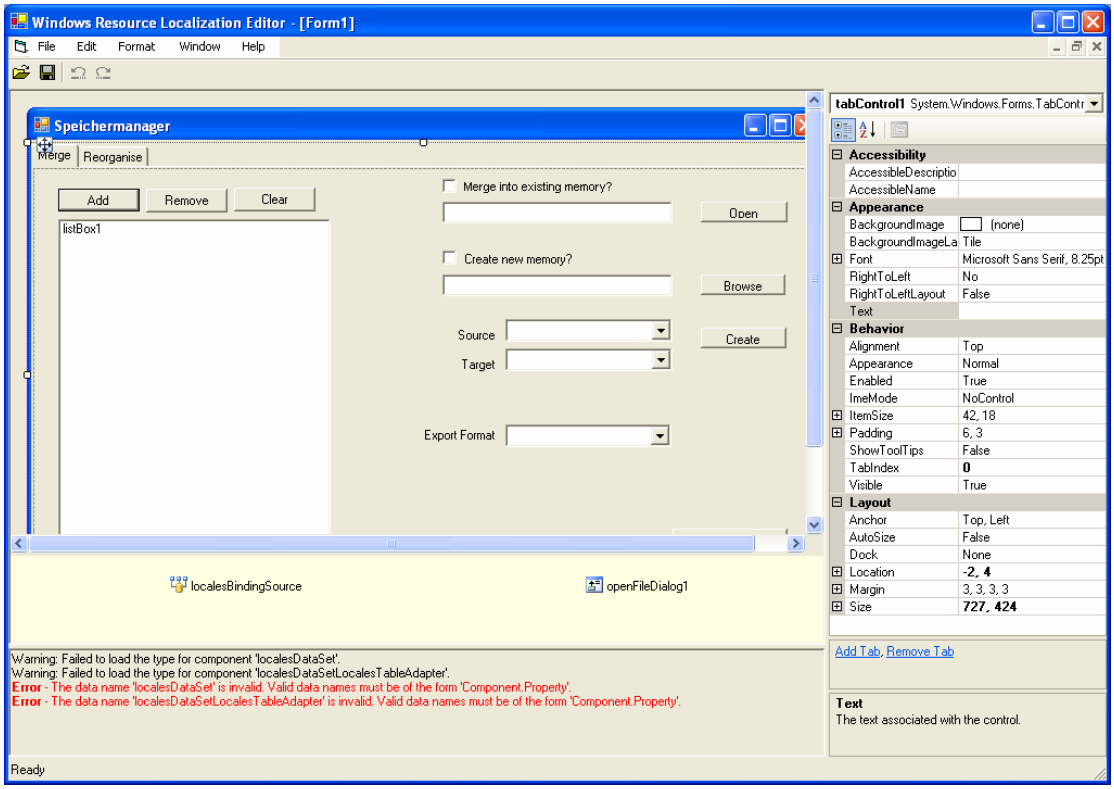
The knowledge I have gained about the .NET framework, Visual Studio, and XML has been great and I hope to use these skills in my future endeavours.

I was pleased with the way I interacted with Xerox employees. The communication process was eased due to the fact I'd worked with most of the people involved during my industrial placement. Regardless of my prior relationship with them, it is important to maintain a respectful, business-like tone in all contact with outside companies. Ideally I would have preferred to spend more time interacting with translators and technical staff, but due to the production demands placed on the staff, as well as geography – I couldn't feasibly travel from Leeds to Welwyn Garden City on a regular basis – this was not possible. I would have preferred to use an iterative development methodology so that I could get regular feedback on the development.

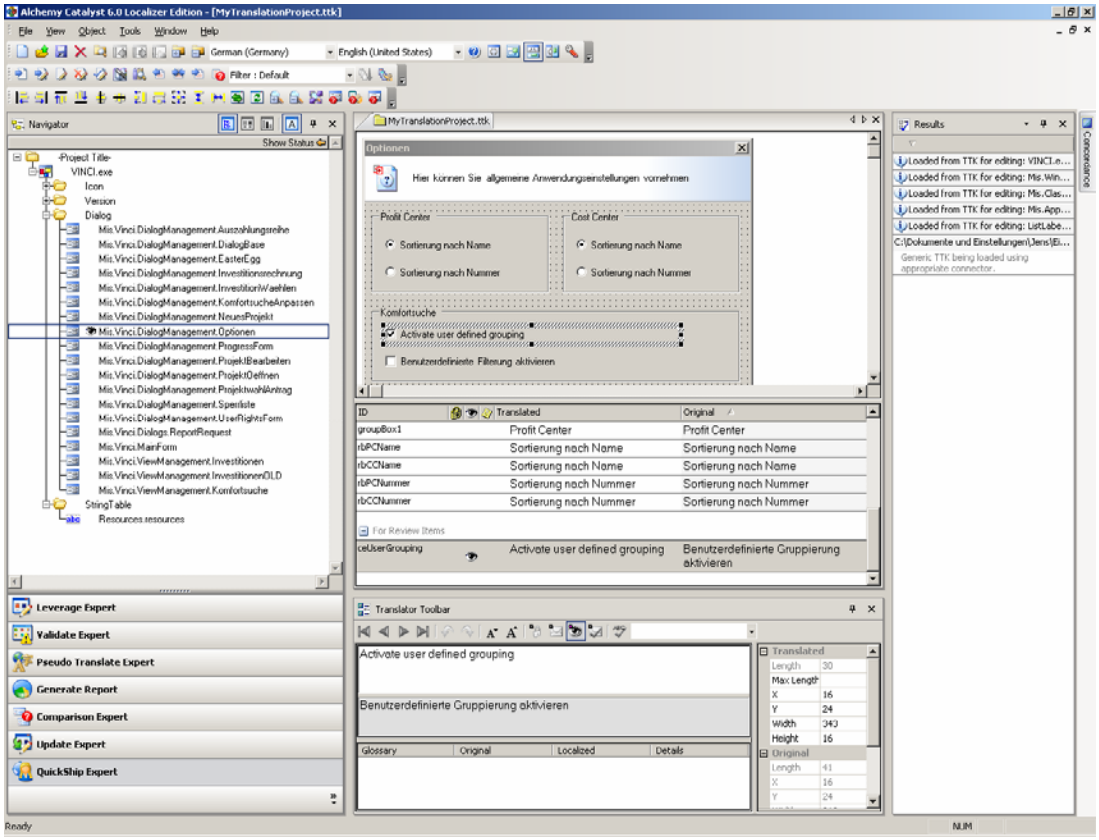
The report writing process is quite arduous and should not be undertaken lightly. I would encourage future students to read some earlier project reports to try and identify the correct tone and style for their report. Organising the structure of the report and writing the appendices of the report is a particularly dull process and therefore will no doubt take a long time. It is best not to underestimate the time necessary for this.

Appendix B – Software Localisation Tools

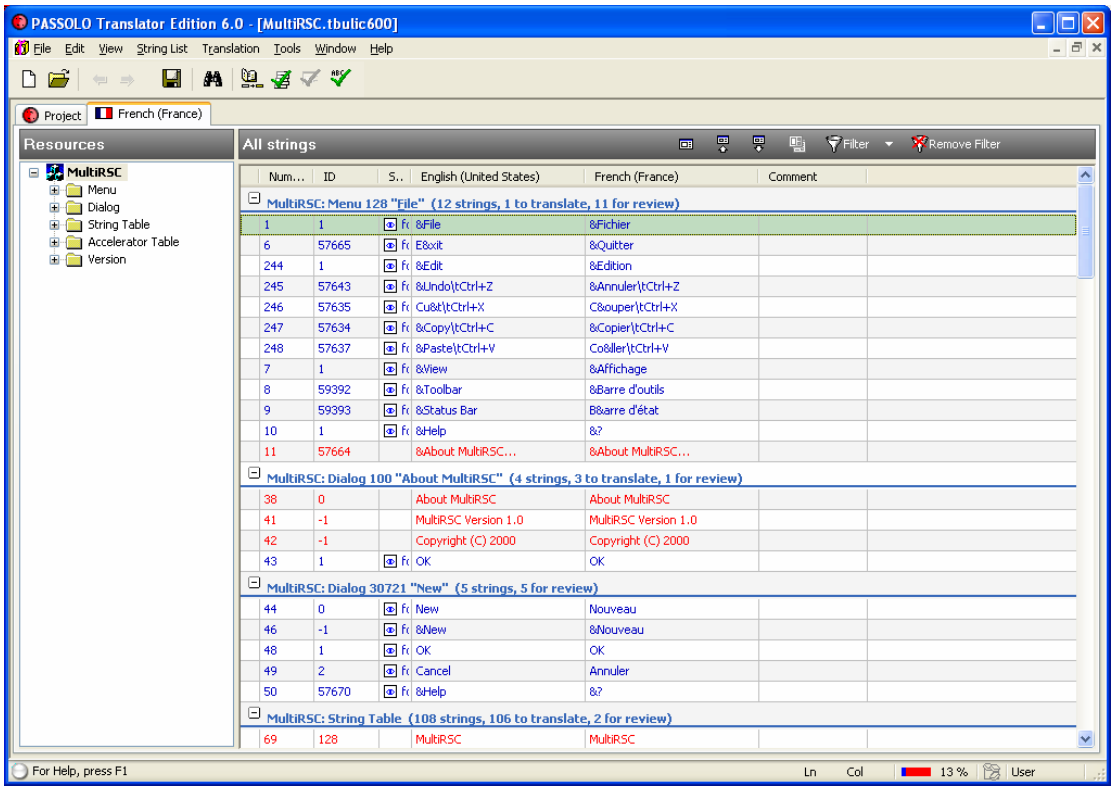
Windows Resource Localisation Editor



Alchemy Catalyst



PASSOLO


PASSOLO Translator Edition 6.0 - [MultiRSC.tbulic600]
File Edit View String List Translation Tools Window Help
Project
French (France)
Resources
All strings
Filter
Remove Filter
MultiRSC
Menu
Dialog
String Table
Accelerator Table
Version
Num... ID S.. English (United States) French (France) Comment
MultiRSC: Menu 128 "File" (12 strings, 1 to translate, 11 for review)
1 1 &File &Fichier
6 57665 E&xit &Quitter
244 1 &Edit &Edition
245 57643 &Undo\tCtrl+Z &Annuler\tCtrl+Z
246 57635 Cu&t\tCtrl+X C&ouper\tCtrl+X
247 57634 &Copy\tCtrl+C &Copier\tCtrl+C
248 57637 &Paste\tCtrl+V Co&ller\tCtrl+V
7 1 &View &Affichage
8 59392 &Toolbar &Barre d'outils
9 59393 &Status Bar B&arre d'état
10 1 &Help &?
11 57664 &About MultiRSC... &About MultiRSC...
MultiRSC: Dialog 100 "About MultiRSC" (4 strings, 3 to translate, 1 for review)
38 0 About MultiRSC About MultiRSC
41 -1 MultiRSC Version 1.0 MultiRSC Version 1.0
42 -1 Copyright (C) 2000 Copyright (C) 2000
43 1 OK OK
MultiRSC: Dialog 30721 "New" (5 strings, 5 for review)
44 0 New Nouveau
46 -1 &New &Nouveau
48 1 OK OK
49 2 Cancel Annuler
50 57670 &Help &?
MultiRSC: String Table (108 strings, 106 to translate, 2 for review)
69 128 MultiRSC MultiRSC
For Help, press F1
Ln
Col
13 %
User

Appendix C – Field Research Notes

Interview Summary

Translator

General Thoughts on TRADOS

Generally useful

Generally good for terminology consistency provided memories are well maintained

For software translation -Limited information in memory about where a term appears in a UI

Types of errors encountered in software translation

- Mainly problems of context
- -Space issues – seeing the screen can benefit here also as seeing the context can show how best to drop some superfluous information i.e. one has already selected Paper Tray in the previous screen, therefore the next screen's heading can drop this bit of information and instead of saying "Paper Tray Attributes" just"Attributes" will suffice), mistranslations that arise from unclear or misleading source text

Improvements to TRADOS

Attaching an ID to terms in the TM so they could be linked to a UI simulation

Annotation is theoretically a good idea, though not sure how practical, if one had to annotate every change made.

Perhaps a report could be generated for future reference.

Would visualisation be a benefit?

Yes, it would make life a lot easier.

Thoughts on other tools compared to TRADOS

Havent used any

Interview Summary

Translator

General Thoughts on TRADOS

-Easy to use

-Overpriced

-Interface with MS Word doesn't work well

-Spell checker is very basic and only contains simple words – little use for German as it uses many compound words

-"Translate to Fuzzy" option is not always reliable

Types of errors encountered in software translation

-Basic errors like spelling mistakes

-Problems of terminology – incorrect terminology for this particular software or screen

Improvements to TRADOS

A WYSWYG mode

Improved spell checker

Improved Word interface – doesn't break down on segments in tables.

Would visualisation be a benefit?

Yes, it would help to deal with terminology problems like those mentioned before.

Thoughts on other tools compared to TRADOS

Has used Nero Across & Catalyst – both of these have a WYSWYG mode but far less user friendly and straight forward compared to TRADOS

Interview Summary

Translator

General Thoughts on TRADOS

Generally fine

Limited flexibility on how segments/paragraphs can be moved around, combined or deleted

Types of errors encountered in software translation

Style – translations not right for this software

Improvements to TRADOS

A verifier that will tell you if you have missed out translating a segment, or accidentally copied the source across to the target – Because there is an option to copy source, it is easily done.

Would being able to annotate translations for future projects be of benefit?

Yes I would

Would visualisation be a benefit?

Definitely, it’s the main problem we face.

Thoughts on other tools compared to TRADOS

Catalyst is useful for context but it’s an expensive add on when you already have TRADOS.

Interview Summary

Translator

General Thoughts on TRADOS

Generally very stable and easy to use

Preview function for HTML is very useful

Types of errors encountered in software translation

-Typos – Down to translator expertise

-Truncations – exceeded bounding boxes

-Messages incorrect in context – what messages are the ones being translated related to

Improvements to TRADOS

Being able to preview all file types when translating.

Would being able to annotate translations for future projects be of benefit?

Potentially yes, there have been occasions where I've wondered what reasoning a translator used when using a certain translation.

Would visualisation be a benefit?

Absolutely!

Thoughts on other tools compared to TRADOS

XGTS – useful for bounding box checking, but slow and bad TM Management

Interview Summary

Technical Staff

- Level of pre processing on files dependent on the file type – some files will enter TRADOS with the correct data available for translation with no preparation. Others require a change of file encoding, tag names changed etc.
- .resx files pre processing time depends on whether we have been sent the coordinate property data as part of the file. If so, that needs blocking off.
- Have looked to automate file preparation for some file types.
- Any tools developed to automate preparation would be welcomed. Would have to be quicker than doing the work manually and fit into the way we work.
- TRADOS is largely good at handling different file types. It's lack of Catalyst-like functionality has been a topic of discussion but it's the best all round tool.

Observation Summary

8/12/06

- Translator viewed for an hour

- The translator was working on a software project, approximately 4000 words as well as checking pre-translated 100% matches to be translated using TRADOS.
- The translator had been provided word count data in advance to help decide whether they had time to carry out the translation
- Used a variety of shortcut keys to speed up work – eg. Ctrl-Alt-PageDn for Open Next/No 100%
- Often consulted reference pack provided by terminologist when they were having issues with a translation – provided as a Word document but printed out by the translator.
- Spent some time cycling through potential translations in translators workbench
- Took some fuzzy translation matches from memory and adapted them for this translation.
- Made some changes to 100% matches – they weren't appropriate for this particular part of the software.
- Translated approximately 300 words in the time spent observing.

Appendix D – Requirements Specification provided to Xerox prior to meeting

Requirements – .resx Visualisation

- Tool to prepare resx files for visualisation plugin and to return the file to it's original state post translation

- Visualisation of .resx files
  - Visualise source and target versions of software whilst viewing the appropriate .TTX file in TRADOS TagEditor.
  - The open segment being translated will be highlighted.
  - Refreshes the view after changes.
  - Reacting to Trados commands - opening new segments, translate to fuzzy etc.

- Bounding Box checking
  - Alert the user that a text bounding box has been exceeded when they close a segment and when exiting the program. The duplicated hotkey will be displayed in red.

- Hotkey checking
  - Alert the user that a hotkey has been duplicated when they close a segment and when exiting the program. The duplicated hotkey will be displayed in red.

- Documentation
  - User manual
  - Programmers reference – A guide to the structure of the code.

Figure 1 – Processing Tool

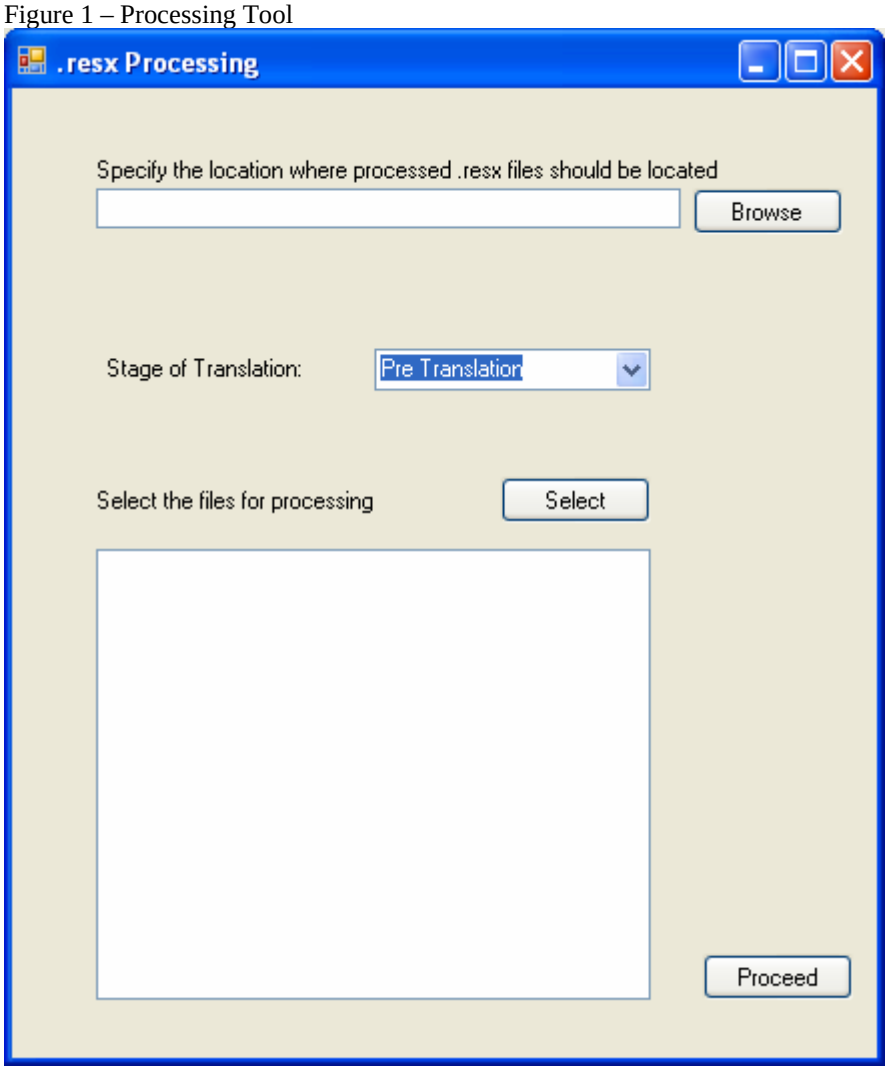


Figure 2 – Visualisation – source view

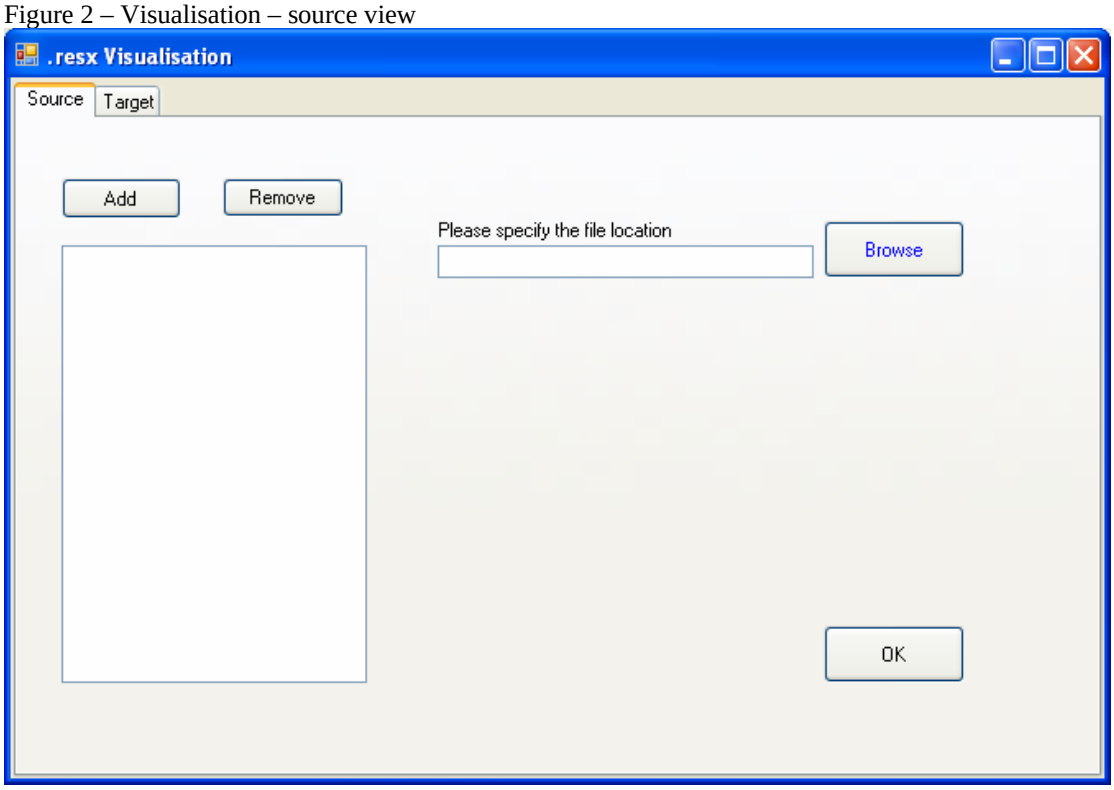
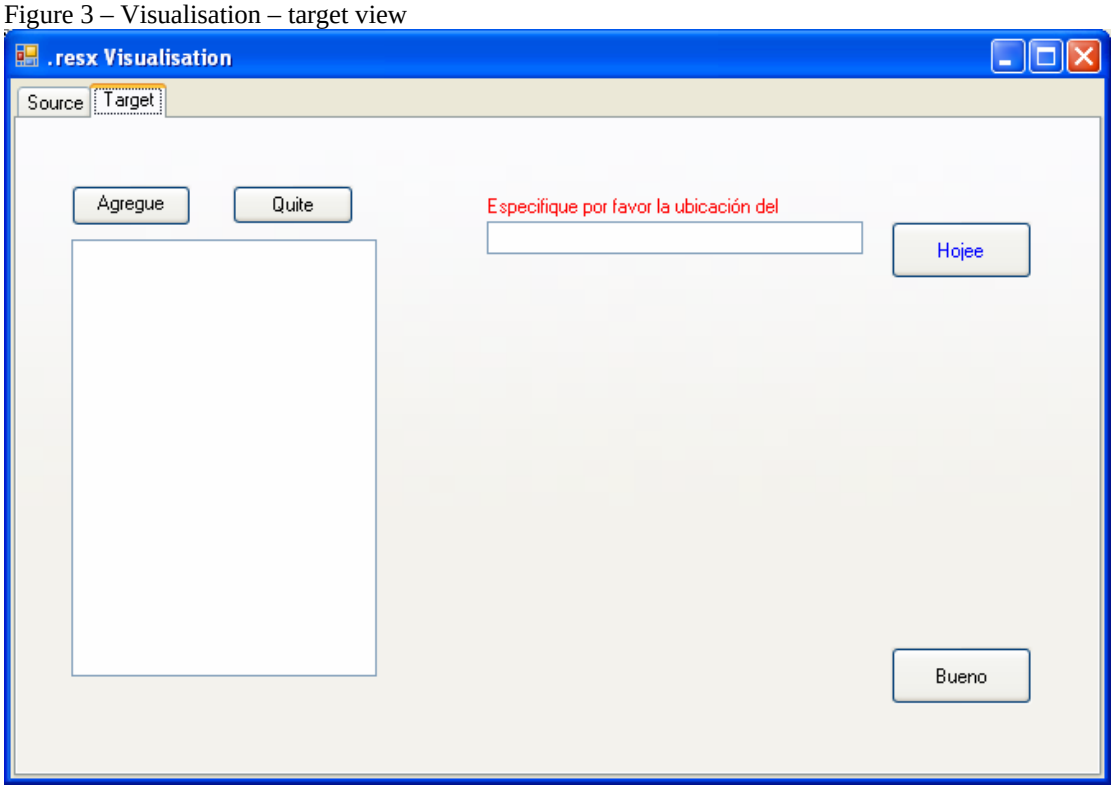


Figure 3 – Visualisation – target view



Requirements Spec – Comment Report and Viewer

- Generate an HTML report from the TTX Comments file containing the source text, translated text and the comment.
- Viewer – a pop up window which presents the user of a future project with relevant comments to their translation, if the text appears in their TTX file.

Appendix E – Documentation

User Guide – Resx Preparation Tool

This software makes changes to the tag information in the source .resx file so that property information other than text is unavailable for translation.

Requirements

Microsoft Windows 2000/XP
.Net Framework 2.0 + J# redistributable package
TRADOS 6.5 or later with license

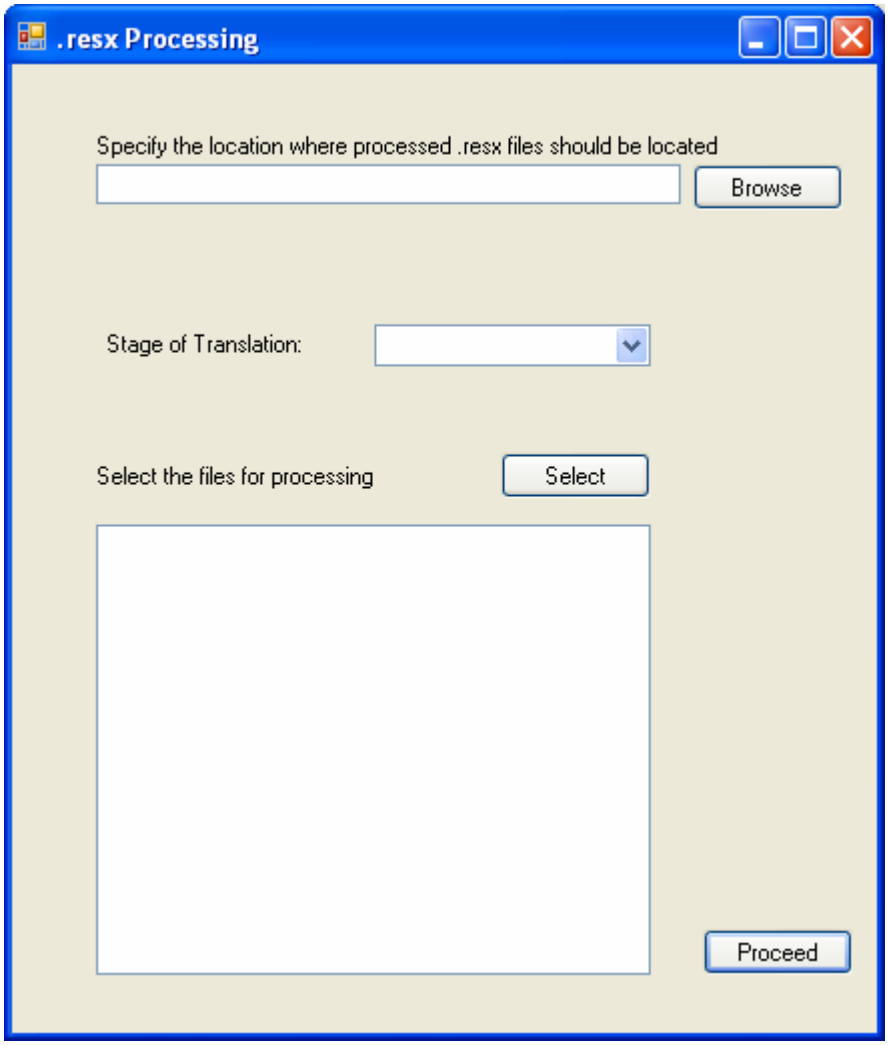
Installation

Double click the ResxVisualisation.msi file
Follow the instructions, click next
Specify the path you would like the software installed to

Running the software

Run the visualization tool – either from the resxpreparation.exe file in the program folder or from the Resx Preparation start menu folder.

The following screen will appear:



To prepare some RESX files, specifiy where you would like the prepared files to be placed. Select the files you wish to prepare, and select the “Pre-Translation” option from the “Stage of Translation” drop down box. Press Ok.

The return RESX files to their original structural state after translation, specify where you would like the files to be placed. Select the files you wish to convert, and select the “Post Translation” option from the “Stage of Translation” drop down box. Press Ok.

User Guide – Resx Visualisation

Requirements

Microsoft Windows 2000/XP
.Net Framework 2.0 + J# redistributable package
TRADOS 6.5 or later with license

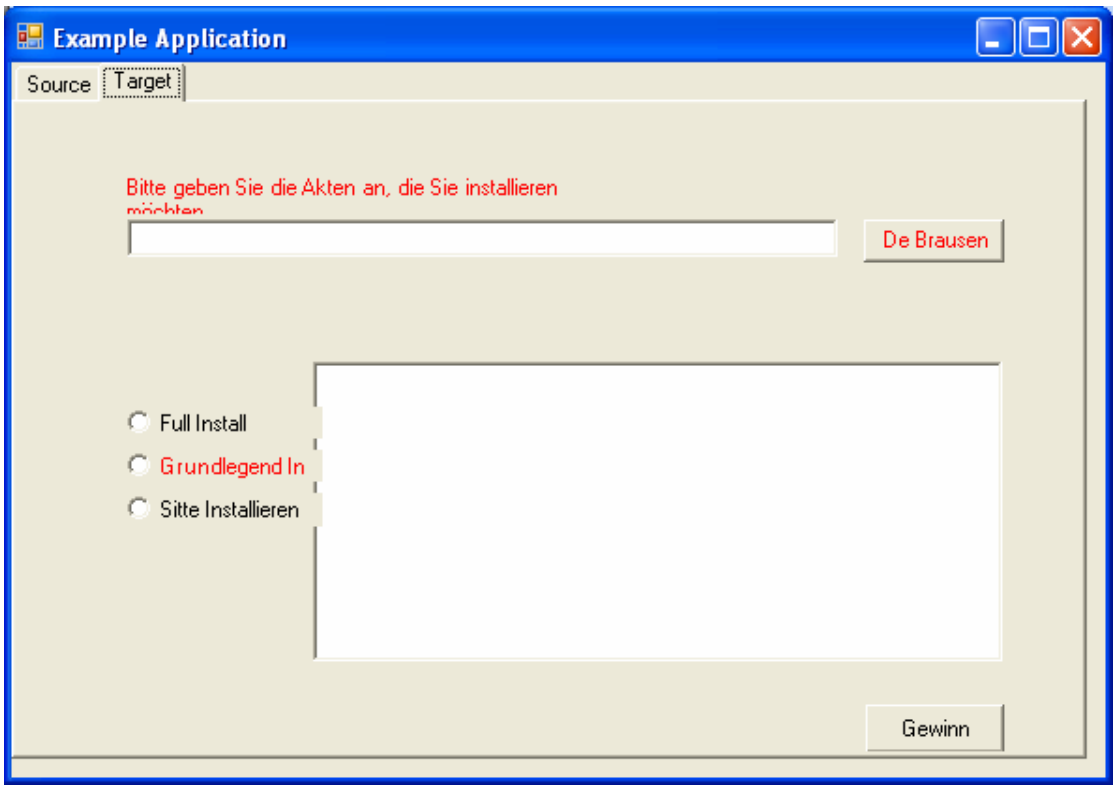
Installation

Double click the ResxVisualisation.msi file
Follow the on screen instructions – the only particular issue you have to deal with here is deciding the path at which you want the software to be installed.

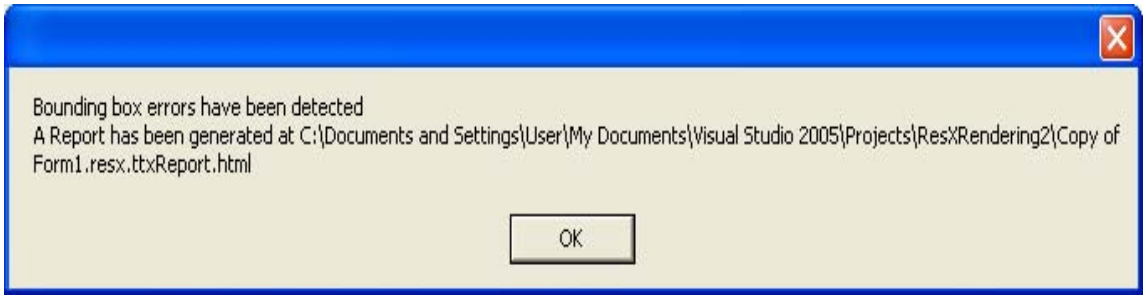
Running the software

Open TRADOS and the .resx.TTX file you would like to translate
Run the visualization tool – either from the resxvisualisation.exe file in the program folder or from the Resx visualization start menu folder.

You should see a window with 2 tabs – source and target – similar to the one below.



Bounding Box Checking
If there were any bounding box errors detected a window like the one below will be displayed


Bounding box errors have been detected
A Report has been generated at C:\Documents and Settings\User\My Documents\Visual Studio 2005\Projects\ResXRendering2\Copy of Form1.resx.ttxReport.html
OK

The text from these will be displayed in red in the target visualization widow. An HTML report is also generated to show the source and target text, as well as the size of the bounding box and how much it has been exceeded by.

If hotkeys are duplicated their text will be displayed in blue. An HTML report will be generated.

User Guide – Comment Report Generation

Requirements
Microsoft Windows 2000/XP
.Net Framework 2.0 + J# redistributable package
TRADOS 6.5 or later with license

Installation
Double click the ResxVisualisation.msi file
Follow the on screen instructions – the only particular issue you have to deal with here is deciding the path at which you want the software to be installed.

Running the software
Right click on a TTX.Comments file and select “Generate comment report” – The TTX file it is associated must be in the same directory. A report like the following will be generated:

User Guide – Comment Viewer

Requirements

Microsoft Windows 2000/XP
.Net Framework 2.0 + J# redistributable package
TRADOS 6.5 or later with license

Installation

Double click the ResxVisualisation.msi file
Follow the on screen instructions – the only particular issue you have to deal with here is deciding the path at which you want the software to be installed.

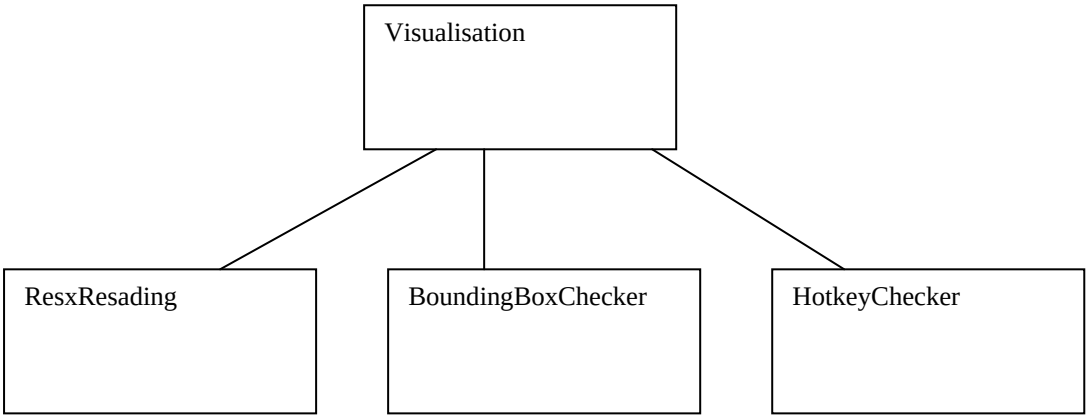
Running the software

Open TRADOS and the .resx.TTX file you would like to translate
Run the comment viewer tool – either from the TTXcommentviewer.exe file in the program folder or from the TTX comment viewer start menu folder.

TTX Comment Viewer
Source Text
Target Text
Comment
Browse
Brausen
Heres a comment

Developers Guide

Visualisation



-Visualisations constructor gets the application and current document object, and calls the ResxReading object to read the TTX file..

ResXReading
All reading in this class is performed with a StreamReader
locateContainer() – looks for the main form and gets its size and name using regular expressions. Adds source and tabPages to a form of its size.
lookForSubContainers() – looks for TabControls and calls their creation method
searchContents() – reads the file looking for the pattern *.Type. If found, it reads its value tag and calls the relevant creation method
newX() – creates the relevant component in the correct array, and makes a copy into the target component array – eg. sourceButtons[buttonCount] and targetButtons[buttonCount]. Deals with any special properties
newComponent() – called by the newX methods to deal with like properties eg. Size, location.
All controls are added to ArrayLists – sourceControls or targetControls

BoundingBoxChecker
Supplied the ArrayLists of source and target controls
CheckBoundingBox(int controlNumber) – uses the TextRenderer library to measure the size of the text and compares it against the source controls size property. If it has exceeded its bounding it is added to the exceededControls list
generateHTMLReport(String path) – creates an HTML file with details of all the exceeded controls.

HotKeyChecker
Supplied the ArrayList of target ToolStripMenuItems
Creates a string array of hotkeys
checkForDuplications() – compares each string in the hotkey array against each other. If a match occurs, the UI components they represent are added to an array of duplicated controls.
generateHTMLReport() –creates an HTML file with details of all the duplicated controls.

ResXPrep Tool

ResXPrep UI – This class is mainly Visual Studio Form Generated code. On pressing ok, an event is fired calling the process or PostProcess files methods from ResxprepControl

ResxPrepControl

processFiles() – Reads each file, creating a new file in the destination directory, writes each line of the old file to the new one, changing the value tag to xvalue if it finds that the name.type data is referring to something that isnt Text

postProcessFiles() - Reads each file, creating a new file in the destination directory, writes each line of the old file to the new one, changing the value tag to xvalue if it finds that the name.type data is referring to something that isnt Text

Also has methods to add or remove files from the filelist.

TTXCommentReader

The constructor takes the filename supplied and uses the DirectoryInfo object to identify what directory the file in, so it can create the HTML report file. It then begins reading the comment file received using an XMLTextReader. If it finds a "Segment" tag, it calls the readComment() method

readComment() – Using the same XMLTextReader as the constructor, looks Start Offset, End Offset, Start Paragraph, End Paragraph and the comment for this segment. Calls the getSourceText() method and then the addCommentToReport() method

getSourceText() – Opens the TTX file the comment relates to, reads to the line number from readComment, then looks for a TU tag, and takes its source and target text from the TUV tags.

TTXCommentViewer

TTXCommentViewerUI – This class is mainly Visual Studio form generated code

displayCommentGrid() – searches through the ArrayList of Comments, adding them to the dataGridView

keyPress() – an event that refreshes the dataGridView if the user presses shift+r.

TTXReaderControl

identifyCommentFiles() – searches the directory of the TTX file for HTML files.

readComments() – confirms whether the HTML file is a comment file, then reads each line taking the source and target text along with the comment.

readTTXFile() – takes a comment, reads the file comparing the source and target text supplied against each translation unit. If found. A new comment object is created with this information in it, which is then added to the commentList ArrayList.

Comment

An object that contains source text, translated text and the comment. Also contains the get methods required. Was created so that the comments could easily be added to the dataGridView.

Appendix F – Testing Results and Sample Files

First Run Through

File	Visualised Correctly	Correct Bounding Box Checking	Correct Hotkey checking	Correct Refreshing
1	Y	Y	Y	Y
2	N	Y	Y	Y
3	N	Y	Y	Y
4	Y	Y	Y	Y

Second Run Through

File	Visualised Correctly	Correct Bounding Box Checking	Correct Hotkey checking	Correct Refreshing
1	Y	Y	Y	Y
2	Y	Y	Y	Y
3	Y	Y	Y	Y
4	Y	Y	Y	Y

Example File Used

```
<?xml version="1.0" encoding="utf-8"?>
<root>
  <!--
    Microsoft ResX Schema

    Version 2.0

    The primary goals of this format is to allow a simple XML format
    that is mostly human readable. The generation and parsing of the
    various data types are done through the TypeConverter classes
    associated with the data types.

    Example:

    ... ado.net/XML headers & schema ...
    <resheader name="resmimetype">text/microsoft-resx</resheader>
    <resheader name="version">2.0</resheader>
    <resheader name="reader">System.Resources.ResXResourceReader,
System.Windows.Forms, ...</resheader>
    <resheader name="writer">System.Resources.ResXResourceWriter,
System.Windows.Forms, ...</resheader>
    <data name="Name1"><value>this is my long string</value><comment>this
is a comment</comment></data>
    <data name="Color1" type="System.Drawing.Color,
System.Drawing">Blue</data>
    <data name="Bitmap1" mimetype="application/x-
microsoft.net.object.binary.base64">
```

```
        <value>[base64 mime encoded serialized .NET Framework
object]</value>
    </data>
    <data name="Icon1" type="System.Drawing.Icon, System.Drawing"
mimetype="application/x-microsoft.net.object.bytearray.base64">
        <value>[base64 mime encoded string representing a byte array form
of the .NET Framework object]</value>
        <comment>This is a comment</comment>
    </data>

    There are any number of "resheader" rows that contain simple
    name/value pairs.

    Each data row contains a name, and value. The row also contains a
    type or mimetype. Type corresponds to a .NET class that support
    text/value conversion through the TypeConverter architecture.
    Classes that don't support this are serialized and stored with the
    mimetype set.

    The mimetype is used for serialized objects, and tells the
    ResXResourceReader how to depersist the object. This is currently not
    extensible. For a given mimetype the value must be set accordingly:

    Note - application/x-microsoft.net.object.binary.base64 is the format
    that the ResXResourceWriter will generate, however the reader can
    read any of the formats listed below.

    mimetype: application/x-microsoft.net.object.binary.base64
    value   : The object must be serialized with
            :
System.Runtime.Serialization.Formatters.Binary.BinaryFormatter
            : and then encoded with base64 encoding.

    mimetype: application/x-microsoft.net.object.soap.base64
    value   : The object must be serialized with
            : System.Runtime.Serialization.Formatters.Soap.SoapFormatter
            : and then encoded with base64 encoding.

    mimetype: application/x-microsoft.net.object.bytearray.base64
    value   : The object must be serialized into a byte array
            : using a System.ComponentModel.TypeConverter
            : and then encoded with base64 encoding.
    -->
  <xsd:schema id="root" xmlns=""
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:msdata="urn:schemas-
microsoft-com:xml-msdata">
    <xsd:import namespace="http://www.w3.org/XML/1998/namespace" />
    <xsd:element name="root" msdata:IsDataSet="true">
      <xsd:complexType>
        <xsd:choice maxOccurs="unbounded">
          <xsd:element name="metadata">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="value" type="xsd:string" minOccurs="0"
/>
              </xsd:sequence>
              <xsd:attribute name="name" use="required" type="xsd:string"
/>
              <xsd:attribute name="type" type="xsd:string" />
              <xsd:attribute name="mimetype" type="xsd:string" />
              <xsd:attribute ref="xml:space" />
```

```
            </xsd:complexType>
          </xsd:element>
          <xsd:element name="assembly">
            <xsd:complexType>
              <xsd:attribute name="alias" type="xsd:string" />
              <xsd:attribute name="name" type="xsd:string" />
            </xsd:complexType>
          </xsd:element>
          <xsd:element name="data">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="value" type="xsd:string" minOccurs="0"
msdata:Ordinal="1" />
                <xsd:element name="comment" type="xsd:string" minOccurs="0"
msdata:Ordinal="2" />
              </xsd:sequence>
              <xsd:attribute name="name" type="xsd:string" use="required"
msdata:Ordinal="1" />
              <xsd:attribute name="type" type="xsd:string"
msdata:Ordinal="3" />
              <xsd:attribute name="mimetype" type="xsd:string"
msdata:Ordinal="4" />
              <xsd:attribute ref="xml:space" />
            </xsd:complexType>
          </xsd:element>
          <xsd:element name="resheader">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="value" type="xsd:string" minOccurs="0"
msdata:Ordinal="1" />
              </xsd:sequence>
              <xsd:attribute name="name" type="xsd:string" use="required"
/>
            </xsd:complexType>
          </xsd:element>
        </xsd:choice>
      </xsd:complexType>
    </xsd:element>
  </xsd:schema>
  <resheader name="resmimetype">
    <value>text/microsoft-resx</value>
  </resheader>
  <resheader name="version">
    <value>2.0</value>
  </resheader>
  <resheader name="reader">
    <value>System.Resources.ResXResourceReader, System.Windows.Forms,
Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089</value>
  </resheader>
  <resheader name="writer">
    <value>System.Resources.ResXResourceWriter, System.Windows.Forms,
Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089</value>
  </resheader>
  <assembly alias="mscorlib" name="mscorlib, Version=2.0.0.0,
Culture=neutral, PublicKeyToken=b77a5c561934e089" />
  <data name="label1.AutoSize" type="System.Boolean, mscorlib">
    <value>True</value>
  </data>
  <assembly alias="System.Drawing" name="System.Drawing, Version=2.0.0.0,
Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a" />
  <data name="label1.Font" type="System.Drawing.Font, System.Drawing">
```

```
    <value>Tempus Sans ITC, 15.75pt</value>
  </data>
  <data name="label1.Location" type="System.Drawing.Point, System.Drawing">
    <value>331, 39</value>
  </data>
  <data name="label1.Size" type="System.Drawing.Size, System.Drawing">
    <value>90, 27</value>
  </data>
  <data name="label1.TabIndex" type="System.Int32, mscorlib">
    <value>0</value>
  </data>
  <data name="label1.Text" xml:space="preserve">
    <value>Test Text</value>
  </data>
  <data name="&gt;&gt;label1.Name" xml:space="preserve">
    <value>label1</value>
  </data>
  <data name="&gt;&gt;label1.Type" xml:space="preserve">
    <value>System.Windows.Forms.Label, System.Windows.Forms,
Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089</value>
  </data>
  <data name="&gt;&gt;label1.Parent" xml:space="preserve">
    <value>$this</value>
  </data>
  <data name="&gt;&gt;label1.ZOrder" xml:space="preserve">
    <value>7</value>
  </data>
  <data name="button1.Font" type="System.Drawing.Font, System.Drawing">
    <value>Snap ITC, 12pt</value>
  </data>
  <data name="button1.Location" type="System.Drawing.Point,
System.Drawing">
    <value>364, 234</value>
  </data>
  <data name="button1.Size" type="System.Drawing.Size, System.Drawing">
    <value>129, 42</value>
  </data>
  <data name="button1.TabIndex" type="System.Int32, mscorlib">
    <value>1</value>
  </data>
  <data name="button1.Text" xml:space="preserve">
    <value>Test Button</value>
  </data>
  <data name="&gt;&gt;button1.Name" xml:space="preserve">
    <value>button1</value>
  </data>
  <data name="&gt;&gt;button1.Type" xml:space="preserve">
    <value>System.Windows.Forms.Button, System.Windows.Forms,
Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089</value>
  </data>
  <data name="&gt;&gt;button1.Parent" xml:space="preserve">
    <value>$this</value>
  </data>
  <data name="&gt;&gt;button1.ZOrder" xml:space="preserve">
    <value>6</value>
  </data>
  <data name="checkBox1.AutoSize" type="System.Boolean, mscorlib">
    <value>True</value>
  </data>
  <data name="checkBox1.Location" type="System.Drawing.Point,
System.Drawing">
```

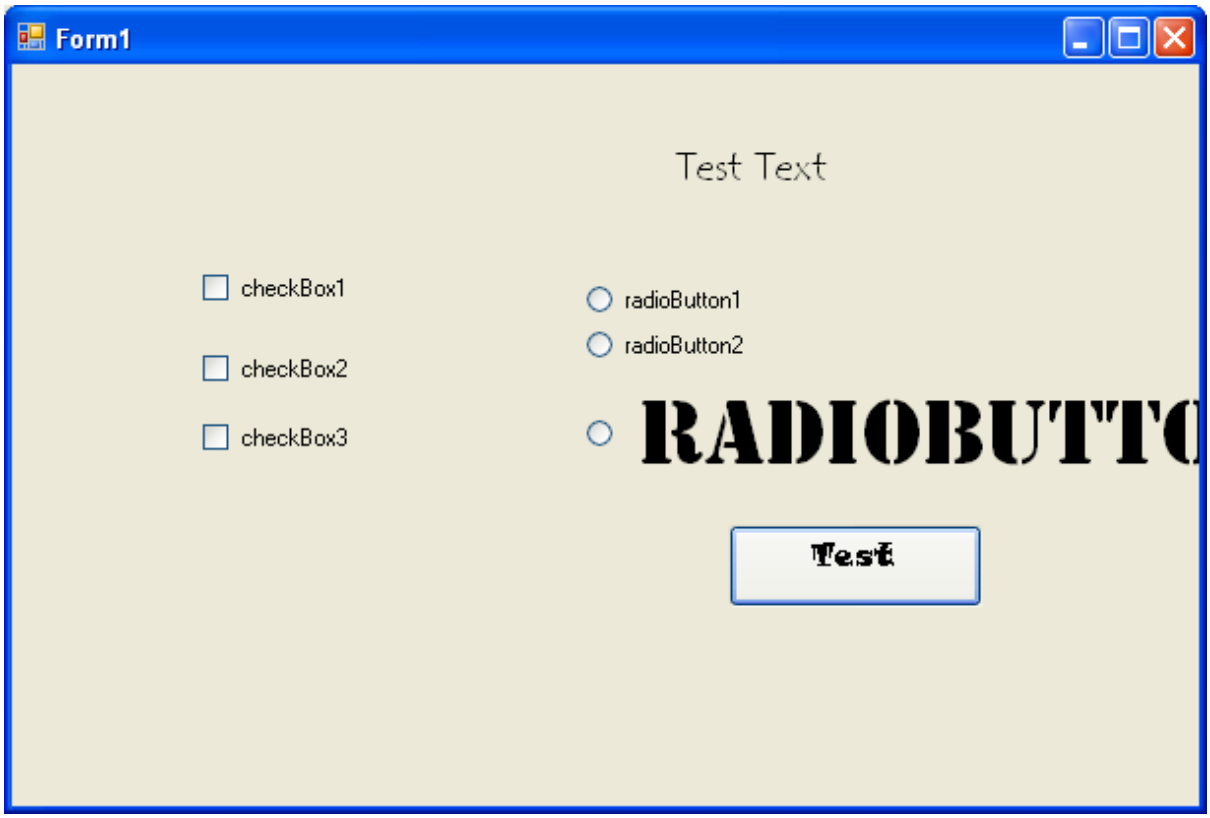
```
    <value>97, 106</value>
  </data>
  <data name="checkBox1.Size" type="System.Drawing.Size, System.Drawing">
    <value>80, 17</value>
  </data>
  <data name="checkBox1.TabIndex" type="System.Int32, mscorlib">
    <value>2</value>
  </data>
  <data name="checkBox1.Text" xml:space="preserve">
    <value>checkBox1</value>
  </data>
  <data name="&gt;&gt;checkBox1.Name" xml:space="preserve">
    <value>checkBox1</value>
  </data>
  <data name="&gt;&gt;checkBox1.Type" xml:space="preserve">
    <value>System.Windows.Forms.CheckBox, System.Windows.Forms,
Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089</value>
  </data>
  <data name="&gt;&gt;checkBox1.Parent" xml:space="preserve">
    <value>$this</value>
  </data>
  <data name="&gt;&gt;checkBox1.ZOrder" xml:space="preserve">
    <value>5</value>
  </data>
  <data name="checkBox2.AutoSize" type="System.Boolean, mscorlib">
    <value>True</value>
  </data>
  <data name="checkBox2.Location" type="System.Drawing.Point,
System.Drawing">
    <value>97, 147</value>
  </data>
  <data name="checkBox2.Size" type="System.Drawing.Size, System.Drawing">
    <value>80, 17</value>
  </data>
  <data name="checkBox2.TabIndex" type="System.Int32, mscorlib">
    <value>3</value>
  </data>
  <data name="checkBox2.Text" xml:space="preserve">
    <value>checkBox2</value>
  </data>
  <data name="&gt;&gt;checkBox2.Name" xml:space="preserve">
    <value>checkBox2</value>
  </data>
  <data name="&gt;&gt;checkBox2.Type" xml:space="preserve">
    <value>System.Windows.Forms.CheckBox, System.Windows.Forms,
Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089</value>
  </data>
  <data name="&gt;&gt;checkBox2.Parent" xml:space="preserve">
    <value>$this</value>
  </data>
  <data name="&gt;&gt;checkBox2.ZOrder" xml:space="preserve">
    <value>4</value>
  </data>
  <data name="checkBox3.AutoSize" type="System.Boolean, mscorlib">
    <value>True</value>
  </data>
  <data name="checkBox3.Location" type="System.Drawing.Point,
System.Drawing">
    <value>97, 182</value>
  </data>
  <data name="checkBox3.Size" type="System.Drawing.Size, System.Drawing">
```

```
    <value>80, 17</value>
  </data>
  <data name="checkBox3.TabIndex" type="System.Int32, mscorlib">
    <value>4</value>
  </data>
  <data name="checkBox3.Text" xml:space="preserve">
    <value>checkBox3</value>
  </data>
  <data name="&gt;&gt;checkBox3.Name" xml:space="preserve">
    <value>checkBox3</value>
  </data>
  <data name="&gt;&gt;checkBox3.Type" xml:space="preserve">
    <value>System.Windows.Forms.CheckBox, System.Windows.Forms,
Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089</value>
  </data>
  <data name="&gt;&gt;checkBox3.Parent" xml:space="preserve">
    <value>$this</value>
  </data>
  <data name="&gt;&gt;checkBox3.ZOrder" xml:space="preserve">
    <value>3</value>
  </data>
  <data name="radioButton1.AutoSize" type="System.Boolean, mscorlib">
    <value>True</value>
  </data>
  <data name="radioButton1.Location" type="System.Drawing.Point,
System.Drawing">
    <value>292, 111</value>
  </data>
  <data name="radioButton1.Size" type="System.Drawing.Size,
System.Drawing">
    <value>85, 17</value>
  </data>
  <data name="radioButton1.TabIndex" type="System.Int32, mscorlib">
    <value>5</value>
  </data>
  <data name="radioButton1.Text" xml:space="preserve">
    <value>radioButton1</value>
  </data>
  <data name="&gt;&gt;radioButton1.Name" xml:space="preserve">
    <value>radioButton1</value>
  </data>
  <data name="&gt;&gt;radioButton1.Type" xml:space="preserve">
    <value>System.Windows.Forms.RadioButton, System.Windows.Forms,
Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089</value>
  </data>
  <data name="&gt;&gt;radioButton1.Parent" xml:space="preserve">
    <value>$this</value>
  </data>
  <data name="&gt;&gt;radioButton1.ZOrder" xml:space="preserve">
    <value>2</value>
  </data>
  <data name="radioButton2.AutoSize" type="System.Boolean, mscorlib">
    <value>True</value>
  </data>
  <data name="radioButton2.Location" type="System.Drawing.Point,
System.Drawing">
    <value>292, 134</value>
  </data>
  <data name="radioButton2.Size" type="System.Drawing.Size,
System.Drawing">
    <value>85, 17</value>
```

```
  </data>
  <data name="radioButton2.TabIndex" type="System.Int32, mscorlib">
    <value>6</value>
  </data>
  <data name="radioButton2.Text" xml:space="preserve">
    <value>radioButton2</value>
  </data>
  <data name="&gt;&gt;radioButton2.Name" xml:space="preserve">
    <value>radioButton2</value>
  </data>
  <data name="&gt;&gt;radioButton2.Type" xml:space="preserve">
    <value>System.Windows.Forms.RadioButton, System.Windows.Forms,
Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089</value>
  </data>
  <data name="&gt;&gt;radioButton2.Parent" xml:space="preserve">
    <value>$this</value>
  </data>
  <data name="&gt;&gt;radioButton2.ZOrder" xml:space="preserve">
    <value>1</value>
  </data>
  <data name="radioButton3.AutoSize" type="System.Boolean, mscorlib">
    <value>True</value>
  </data>
  <data name="radioButton3.Font" type="System.Drawing.Font,
System.Drawing">
    <value>Stencil, 36pt</value>
  </data>
  <data name="radioButton3.Location" type="System.Drawing.Point,
System.Drawing">
    <value>292, 157</value>
  </data>
  <data name="radioButton3.Size" type="System.Drawing.Size,
System.Drawing">
    <value>399, 61</value>
  </data>
  <data name="radioButton3.TabIndex" type="System.Int32, mscorlib">
    <value>7</value>
  </data>
  <data name="radioButton3.Text" xml:space="preserve">
    <value>radioButton3</value>
  </data>
  <data name="&gt;&gt;radioButton3.Name" xml:space="preserve">
    <value>radioButton3</value>
  </data>
  <data name="&gt;&gt;radioButton3.Type" xml:space="preserve">
    <value>System.Windows.Forms.RadioButton, System.Windows.Forms,
Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089</value>
  </data>
  <data name="&gt;&gt;radioButton3.Parent" xml:space="preserve">
    <value>$this</value>
  </data>
  <data name="&gt;&gt;radioButton3.ZOrder" xml:space="preserve">
    <value>0</value>
  </data>
  <metadata name="$this.Localizable" type="System.Boolean, mscorlib,
Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089">
    <value>True</value>
  </metadata>
  <data name="$this.AutoScaleDimensions" type="System.Drawing.SizeF,
System.Drawing">
    <value>6, 13</value>
```

```
  </data>
  <data name="$this.ClientSize" type="System.Drawing.Size, System.Drawing">
    <value>603, 377</value>
  </data>
  <data name="$this.Text" xml:space="preserve">
    <value>Form1</value>
  </data>
  <data name="&gt;&gt;$this.Name" xml:space="preserve">
    <value>Form1</value>
  </data>
  <data name="&gt;&gt;$this.Type" xml:space="preserve">
    <value>System.Windows.Forms.Form, System.Windows.Forms,
Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089</value>
  </data>
</root>
```

This software looks like the following:



Appendix G – End User Evaluation

Interview Summary

05/04/07

- Does this software fit into your work well?

Yes, I looked at the window on start up, then maximised in whenever I was unsure on a translation for assistance.

Unobtrusive

- Was it easy to pick up how to use the software?

Think so, yes. It just loads and off you go!

- Do you have any suggestions for improvements?

If the visualisation software could tell you what the segment currently open

- In its current state, would you be happy to continue using it?

Yes, although it would be greatly improved with the improvement suggested.

- From the demonstration, did you think that the comment system would be of benefit?

I believe so, but it is difficult to tell without putting it into use over a number of projects.

Other Comments

Queried whether it was only for RESX files.

Felt that the comment functionality would be useful when evaluating new translators. They could make comments and pass the report onto a resourcing specialist.

Interview Summary

06/04/07

- Does this software fit into your work well?

Think so, was certainly useful to have.

- Was it easy to pick up how to use the software?

Yes – Not a lot to learn really!

- Do you have any suggestions for improvements?

Being able to switch off the hotkey and bounding box checking

- In its current state, would you be happy to continue using it?.

I'd prefer it if there was an option to turn off the hotkey checking as it is not always our job to deal with it. But oh the whole it would benefit my work greatly.

- From the demonstration, did you think that the comment system would be of benefit?

Yes – although would like to work with it some more.
The comment system could benefit from being able to categorise the types of comments being made eg. Ones regarding terminology

Observation Summary

8/12/06

- Translator observed for the duration of the test translation (65 minutes)
- Started the visualisation tool with no problems
- Had a few minutes looking at the software and how the pre-translated 100% matches looked in context
- The translator had the visualisation window minimised for the majority, occasionally referring back to it after a few translations.
- Used the Shift-R refresh button
- Didn’t make any new bounding box errors, only had to correct the ones introduced by pre-translation.
- Less use of the terminology reference material than in previous observation

Interview Summary

05/04/07

- Does this software fit into your work well?

Yes, no problems.

- Was it easy to pick up how to use the software?

Yes, I didn't encounter any problems

- Do you have any suggestions for improvements?

If it could tell you exactly where the segment you are translating was on the interface that would be useful.

- In its current state, would you be happy to continue using it?.

I'd prefer it if there was an option to turn off the hotkey checking as it is not always our job to deal with it. But oh the whole it would benefit my work greatly.

Interview Summary

05/04/07

- Does this software fit into your work well?

Yes I was very happy with it

- Was it easy to pick up how to use the software?

Yes, I didn't encounter any problems

- Do you have any suggestions for improvements?

Maybe if you could put the text you are currently viewing in workbench into the visualisation to see how it would look in context. That might be useful.

- In its current state, would you be happy to continue using it?.

Yes, I expect it would benefit by work in future if I continued to use it.

- From the demonstration, did you think that the comment system would be of benefit?

It's a bit difficult to tell from a demonstration but I think it has potential.